AFRL-SR-AR-TR-10-0206

REPORT DOCUMENTATION PAGE

1. REPORT DATE *(DD-MM-YYYY)* 03 FEB 09	2. REPORT TYPE FINAL REPORT	3. DATES COVERED *(From - To)* 01 MAR 07 TO 30 NOV 09

4. TITLE AND SUBTITLE FINITE-TIME PERFORMANCE OF LOCAL SERACH ALGORITHMS: THEORY AND APPLICATION	5a. CONTRACT NUMBER FA9550-07-1-0232
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) DR SHELDON H. JACOBSON	5d. PROJECT NUMBER 2304
	5e. TASK NUMBER DX
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) UNIVERSITY OF ILLINOIS 506 S WRIGHT ST, 364 HENRY ADMIN BLDG URBANA, IL 61801-3620	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFOSR/NL 875 NORTH RANDOLPH STREET SUITE 325, ROOM 3112 ARLINGTON, VA 2203-1768	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE. DISTRIBUTION IS UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Several applications such as a military resource allocation problem, an airborne laser targeting sequence problem, an equal flow on generalized networks and a perimeter screening problem which are all of relevance to the Air Force and Homeland Security will be considered.

20100617296

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON	
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(Include area code)*	

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH

DTIC Data

Purchase Request Number:	FQ8671-0900026
BPN:	F1ATA08224B022
Proposal Number:	06-NM-310
Research Title:	FINITE-TIME PERFORMANCE OF LOCAL SEARCH ALGORITHMS: THEORY AND APPLICATION
Type Submission:	Final Report
Inst. Control Number:	FA9550-07-1-0232P00002
Institution:	UNIVERSITY OF ILLINOIS
Primary Investigator:	Dr. Sheldon H. Jacobson
Invention Ind:	none
Project/Task:	2304D / X
Program Manager:	Donald Hearn

Objective:

This objective of this research is to explore novel approaches and methods to analytically assess the finite-time performance of local search algorithms for large-scale, intractable discrete optimization problems.

Approach:

Threshold analysis considers threshold values for the objective function of a discrete optimization problem, and hence, as these threshold values are incrementally decreased towards the globally optimal objective function value, the performance of the algorithm can be assessed and measured. This project will be focused on 3 approaches: 1) describing the threshold analysis framework for measuring the performance of local search algorithms for discrete optimization problems, 2) introducing and studying how a new design of experiment, termed designed replication, can be used to estimate and compute these performance measures, 3) studying how antithetic and common random number streams, control variates and logistics regression statistical procedures can be used to improve the estimation and application of these performance measures, and 4) studying the application of these performance measures to several local search algorithms such as simulated annealing, genetic algorithms and tabu searches for large-scale, hard discrete optimization problems.

Progress:

Year: 2007 **Month:** 01

Not required at this time.

Year: 2008 **Month:** 02

First result quantifies effectiveness of local search algorithms on discrete oprimization problems by the choice of neighborhood functions. The second result considers a homeland security design and optimization problem termed Sequential Stochastic Security Design problem. which models security and bag-screening operations. The third result introduces a discrete optimization problem framework for obtaining optimal subsets of solutions from large sets of Pareto optimal solutions.

Year: 2009 **Month:** 02

The primary research activities focused on exploring new approaches to analytically assess the finite-time performance of local search algorithms for large-scale, intractable discrete optimization problems (DOPs), and utilizing such models to address homeland security and military problems. Convergence results for local search algorithms applied to DOPs typically require the number of iterations to approach infinity. Finite-time performance results are more useful, but typically are more difficult to obtain. A tradeoff between these two extreme objectives is the finite time performance of such

Progress:

Year: 2009 **Month:** 02

algorithms in visiting near-optimal solutions. To achieve this, the concept of a beta-acceptable solution is used to measure the effectiveness of local search algorithms. To provide a rigorous analysis of this approach, Markov chain state pooling is introduced to obtain estimators for the expected number of iterations to visit such near-optimal solutions. Convergence results for the resulting estimator are reported. Extensive computational results are reported with the Lin-Kernighan-Helsgaun algorithm applied to medium and large traveling salesman problem instances taken from TSPLIB. The results of this research provide a framework for efficiently comparing different local search algorithms in their effectiveness in reaching sub-optimal solutions.

Year: 2009 **Month:** 12 **Final**

This project explored novel approaches and methods to analytically assess the finite-time performance of local search algorithms for large-scale, intractable discrete optimization problems. The results obtained provide finitetime and asymptotic (near globally optima) performance measures for the effectiveness of local search algorithms. These performance measures capture properties of the number of iterations that a local search algorithm requires to visit (for the first time) a solution with objective function value at or below various thresholds. The resulting measures provide both theoretical and estimation tools for analyzing and designing a broad spectrum of local search algorithms for intractable, large-scale, discrete optimization problems, as well as for assessing the appropriateness of specific local search algorithms for particular classes of problems. The primary applications considered for these methodologies included scheduling, circuit, and perimeter screening problems.

FINAL TECHNICAL REPORT

Submitted to:	Donald Hearn, Ph.D.
	Program Manager: Optimization and Discrete Mathematics
	Directorate of Mathematics and Information Sciences
	Air Force Office of Scientific Research
	875 North Randolph Street
	Arlington, VA 22203
Principal Investigator:	Sheldon H. Jacobson, Ph.D.
	Director, Simulation and Optimization Laboratory
	Department of Computer Science
	University of Illinois
	Urbana, IL 61801-2302
	(217) 244-7275 (office)
	shj@illinois.edu
	netfiles.uiuc.edu/shj/www/shj.html
Grant Number:	FA9550-07-1-0232

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The research conducted under this grant focused on the mathematical analysis of local search algorithms for hard, large-scale, discrete optimization problems, multi-criteria post-optimality analysis, and stochastic dynamic optimization problems. The primary technical accomplishments achieved include

i) quantifying the relationship between the size of neighborhood functions and the number of local optima over generic solution spaces that model hard discrete optimization problem landscapes,

ii) designing a methodology to estimate the expected number of iterations to visit a β-acceptable solution using local search algorithms for hard discrete optimization problems,

iii) introducing a discrete optimization framework for obtaining optimal subsets of solutions from large sets of Pareto optimal solutions,

iv) formulating and analyzing the Sequential Stochastic Security Design Problem, which models the screening operations of passengers and carry-on baggage in an aviation security system.

v) identifying a real-time methodology for screening objects that are attempting to enter and harm a secure area under a binary screening paradigm and showing how this methodology can be used to provide insights into the operation and performance of such real-time systems.

vi) introducing a statistical framework for comparing the performance of heuristics for hard discrete optimization problems.

All the accomplishments resulting form the research reported under this grant are documented in several archival journal articles and book chapters. Many of the results have also been presented at national and international conferences, and have won awards for their contribution.

Three Ph.D. dissertations were completed during the period of the grant. Dr. Alexander G. Nikolaev successfully defended and submitted his Ph.D. dissertation "Stochastic Sequential Resource Allocation and Passenger Assignment in Aviation Security Systems " in August 2008. At present, he is a visiting assistant professor at Northwestern University. Dr. Gio K. Kao successfully defended and submitted his Ph.D. dissertation "Two Combinatorial Optimization Problems at the Interface of Computer Science and Operations Research " in August 2008. At present, he is on the technical staff at Sandia National Laboratory. Dr. Adrian J. Lee successfully defended and submitted his Ph.D. dissertation "Optimality, Uncertainty, and Performance of Passenger Screening in Aviation Security Systems" in May 2009. At present, he is a post-doctoral research fellow at the University of Illinois at Chicago, in the Vishwamitra Research Institute.

1. An Analysis of Neighborhood Functions on Generic Solution Spaces

Local search algorithms provide effective and efficient tools for addressing hard discrete optimization problem instances that have very large solution spaces (i.e., exponentially many solutions with respect to the input data size) (Aarts and Lenstra 1997). Local search algorithms iteratively search the solution space in an effort to find near-optimal solutions. Local search algorithms use a neighborhood function that specifies adjacent solutions, and hence, determines how the solution space is traversed. When a local search algorithm (with neighborhood function η) is applied to an instance I of a discrete minimization problem (i.e., a solution space and an objective function over this space), an initial solution ω_0 is constructed and the neighborhood of this solution, $\eta(\omega_0)$, is searched for a solution ω_1 with smaller objective function value than ω_0. Then the neighborhood of this solution, $\eta(\omega_1)$, is then searched for a solution ω_2 with smaller objective function value than ω_1. This process is repeated until a local minimum solution ω_k (for some $k = 1,2,\ldots$) is reached. Since it is often impossible to search an entire solution space for a desirable solution, the neighborhood function is used to focus the search for improving solutions on (typically) small subsets (neighborhoods) of the solution space. In general, the size of a neighborhood is equal to the order of magnitude of the number of solutions in that neighborhood. For example, consider a discrete optimization problem where the solution space is the set of binary vectors $\{0,1\}^n$. Then the k-flip neighborhood function of a solution $\omega = (\omega_1, \omega_2,\ldots, \omega_n) \in \{0,1\}^n$ consists of all binary vectors $\zeta = (\zeta_1,\zeta_2,\ldots,\zeta_n) \in \{0,1\}^n$ that can be obtained from switching k components of ω (i.e., $\Sigma_{i=1,2,\ldots,n} |\omega_i - \zeta_i| = k$). This k-flip neighborhood function has size $O(n^k)$.

The challenge when using local search algorithms is that when a local optimum is reached, it is often not a global optimum. Therefore, the effectiveness of local search algorithms is highly influenced by the choice of the neighborhood function. More specifically, understanding the likelihood that a given neighborhood function has zero (or few) local optima that are not global optima may impact how a local search algorithm is implemented or used. For a neighborhood function with zero (or few) local optima, an upper bound on the time to find an improving solution to any given solution, except global optima, is a product of the time it takes to compute the objective function and the number of solutions in the neighborhood (which are both known). One can then assess whether it is reasonable to run a local search algorithm for additional iterations after a certain quality solution is obtained. This is an advantage for neighborhood functions with no local optima that are not global optima, as compared to neighborhood functions with many such local optima, since for the latter neighborhood functions, it could take a prohibitive (exponential) amount of time to find an improving solution. If the likelihood of having no local optima that are not global optima is known for many different neighborhood functions, then one can determine (after obtaining a certain quality solution) if a larger neighborhood with its longer search time is preferable to reach a particular objective function value (see Jacobson et al. 2005 for a description of one such measure). Therefore, knowing the likelihood that different neighborhood functions have no local optima that are not global optima could be useful in designing a local search algorithm in which the neighborhood function changes as the algorithm progresses. Furthermore, if a neighborhood function has a high likelihood of having zero (or few) local optima that are not global optima, then any local optimum would then have a high likelihood of being a global optimum. This suggests that knowing the likelihood of having no (or few) local optima that are not global optima is helpful in determining the quality of local optima found by a local search algorithm.

Large neighborhood functions are likely to lead to few local optima that are not global optima. Therefore, a local search algorithm using such a neighborhood function is likely to find an improving solution in the neighborhood of a randomly selected solution. However, it generally takes more computing time to search large neighborhoods for an improving solution (Ahuja et al. 2002). This suggests two opposing situations: using large neighborhood functions that are computationally intensive to search but result in few local optima that are not global optima, versus using small neighborhood functions that are computationally efficient to search but result in a large number of local optima that are not global optima. The ideal situation is to create neighborhood functions that are both computationally efficient to search and are sufficiently large such that they result in zero (or few) local optima that are not global optima. However, this is highly unlikely to occur for all instance of a hard discrete optimization problem, assuming P $\neq$ NP. To provide results based on this observation, this research examines how increasing the neighborhood size affects the likelihood that the resulting neighborhood function has no local optima that are not global optima, for generic solution spaces. Such results represent a first step towards providing insights and information on deciding if the reduced number of local optima (that are not global optima) that comes with larger neighborhoods is worth the associated increased search times.

Researchers have begun to examine the question of how to choose an effective neighborhood function. For example, the autocorrelation coefficient of a neighborhood function (Angel and Zissimopoulos 2000) measures the closeness of objective function values of neighboring solutions, by quantifying the continuity of a neighborhood function on a discrete solution space. A high value for the autocorrelation coefficient suggests that the objective function values of neighboring solutions are close, which suggests that the number of local optima may be small. The autocorrelation coefficient can be used to compare two different neighborhood functions for a given problem. Other recent work with neighborhood functions includes results on large-scale (exponentially sized) neighborhood functions (Ahuja et al. 2002), where the size of the neighborhood is defined to be very large with respect to the input data size. However, to make these neighborhood functions practical, algorithms are needed so that they can be searched efficiently. The general consensus is that larger neighborhood functions are attractive since they tend to result in fewer local optima.

There have also been several results reported in the literature on complexity issues associated with neighborhood functions. Johnson et al. (1988) introduce the class of problems PLS and investigate the complexity of finding local optima for neighborhood functions of discrete optimization problems. The class of problems PLS contains all local search problems (discrete optimization problems together with a neighborhood function) in which local optimality can be verified in polynomial time. Jacobson and Solow (1993) and Armstrong and Jacobson (2003) investigate the complexity of finding polynomial time improvement algorithms for discrete optimization problems. A polynomial time improvement algorithm is a polynomial-time algorithm such that given any solution, either another solution can be found with better objective function value or else the algorithm concludes that no such solution exists and the given solution is a global optimum. A polynomial time improvement algorithm exists when a polynomially computable neighborhood function with zero L-locals is available.

Armstrong and Jacobson (2006b) investigate the complexity of finding neighborhood functions with specified desirable properties (such as zero L-locals or polynomial number of L-locals) for NP-hard discrete optimization problems. Armstrong and Jacobson (2006b) introduce the neighborhood transformation and show that MAX Clause Weighted Satisfiability (MCWS), Zero-One Integer Programming (ZOIP), and other NP-hard problems are NPO-complete with respect to neighborhood transformations. There are several implications of these results. In particular, if MCWS or ZOIP has a polynomially computable neighborhood function with zero L-locals, then so does every other problem in NPO. Also, if MCWS or ZOIP has a polynomially computable neighborhood function with a polynomial number of L-locals, then so does every other problem in NPO. Armstrong and Jacobson (2006a) show that other related properties are preserved by neighborhood transformations, such as neighborhood functions with paths to local optima (global optima) of polynomial length.

Several papers in the literature examine the number of L-locals for reasonable neighborhood functions (Tovey 1985) of NP-hard discrete optimization problems. A reasonable neighborhood function is independent of the problem data and has size that is polynomial in the input data size. Vizing (1977) and Savage et al. (1976) independently show that any problem parameter independent neighborhood function of TSP for which all local optima are global optima must be exponentially large, hence there does not exist a reasonable neighborhood function for TSP with zero L-locals. Tovey (1985) shows that every reasonable neighborhood function for Maximum Clique and MAX SAT does not have the smooth property (every strict L-local is a global optimum). Armstrong and Jacobson (2005) strengthen this result by showing that *all* data independent neighborhood functions for MAX 3-SAT do not have the smooth property, except for neighborhood functions that contain all possible solutions for instances with $n \geq 4$ Boolean variables. Rodl and Tovey (1987) also demonstrate that for Maximum Independent Set, there exists a graph (up to the relabeling of the vertices) such that all neighborhood functions of polynomial size have exponentially many local optima.

The purpose of this research is to gain new insight into the number of local optima when using particular neighborhood functions over a *generic solution space*, (i.e., a set of discrete objects and a set of values associated with each object. By design, any discrete optimization can be modeled as a generic solution space; see Fleischer and Jacobson 1999.) In general, a neighborhood function that leads to a small number of local optima that are not global optima is desirable when applying local search algorithms. Moreover, a good neighborhood function should also have a small *diameter* (i.e., the smallest positive integer d such that there exists a path between every pair of solutions with d or fewer neighborhood connections) and a *high correlation* among neighbors (i.e., solutions with objective function values close together will be neighbors of each other). This research examines the relationship between the size of neighborhood functions and the number of local optima. The objective of this research is to provide insights into how to select neighborhood functions, and hence, facilitate the development of new local search algorithms in which the neighborhood function changes as the algorithm progresses. The ultimate (long-term) goal of this research is to develop rules or guidelines when choosing a neighborhood function for discrete optimization problems.

The results reported provide an important first step in examining the number of neighborhood functions with zero L-locals. These results report include the number of neighborhood functions with zero L-locals over a generic solution space that consists of an objective function that is one-to-one (each solution has a unique objective function value in an instance). The hope is that these results will eventually be extended to NP-hard discrete optimization problems, which will begin the process of developing information on solution landscapes (i.e., the number of local optima, relationship among local optima, reachability of local optima) of commonly used neighborhood functions; such information could be helpful when selecting neighborhood functions for such problems.

Several definitions are needed to discuss the results obtained and reported. A *discrete optimization problem* (Ω, f) is formulated to find a solution $\omega \in \Omega$ that optimizes f, where Ω is the (finite) *solution space* and f: $\Omega \to \Re$ is the *objective function* (Garey and Johnson 1979). Without loss of generality, assume that discrete optimization problems are minimization problems. Given a discrete optimization problem (Ω, f), a *neighborhood function* η: $\Omega \to 2^{\Omega}$ maps each element of Ω to a set of elements in Ω such that $\omega \notin \eta(\omega)$ for all $\omega \in \Omega$. Note that this definition states that there does not exist a solution that is in its own neighborhood (this restriction is needed so that a definition for strict local minimum (given below) is well-defined.) Local search algorithms use a neighborhood function to traverse the solution space (Jacobson et al. 1998). Therefore, a *search formulation* is defined to be the three-tuple (Ω, f, η). A neighborhood function η is *symmetric* if $\omega_1 \in \eta(\omega_2)$ implies $\omega_2 \in \eta(\omega_1)$ for all $\omega_1, \omega_2 \in \Omega$. Assume that the neighborhood functions considered in this research are symmetric. Neighborhood functions η_1 and η_2 are *equivalent* if for all $\omega \in \Omega$, $\eta_1(\omega) = \eta_2(\omega)$, while neighborhood functions η_1 and η_2 are *distinct* (or different) if there exists $\omega \in \Omega$ such that $\eta_1(\omega) \neq \eta_2(\omega)$.

For a discrete optimization problem (Ω, f), define a global minimum (maximum) to be a solution $\omega \in \Omega$ such that $f(\omega) \leq (\geq) f(\omega')$ for all $\omega' \in \Omega$. Given a search formulation (Ω, f, η), a solution $\omega \in \Omega$ is a *local minimum* (*maximum*) if $f(\omega) \leq (\geq) f(\omega')$ for all $\omega' \in \eta(\omega)$. Also, for a search formulation (Ω, f, η), a solution $\omega \in \Omega$ is a *strict local minimum* (*maximum*) if $f(\omega) < (>) f(\omega')$ for all $\omega' \in \eta(\omega)$. A solution $\omega \in \Omega$ is a (*strict*) *L-local* if ω is a (strict) local optimum that is not a global optimum.

Neighborhood functions for discrete optimization problems can be described in several ways. Symmetric neighborhood functions can be modeled using an undirected graph $G = (V, E)$ (for undirected graphs, edge (u, v) is equivalent to the edge (v, u)). Given a neighborhood function η for a discrete optimization problem (Ω, f), the *neighborhood graph* (Angel and Zissimopoulos 2000) of η has vertex set $V = \Omega$ and edge set E where $(\omega', \omega) \in E$ if and only if $\omega' \in \eta(\omega)$. Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic* if there exists a bijection (i.e., one-to-one and onto function) ϕ: $V_1 \to V_2$ such that $(u, v) \in E_1$ if and only if $(\phi(u), \phi(v)) \in E_2$ (see Chartrand and Oellermann 1993). A neighborhood function is *r-regular* for some positive integer r if the corresponding neighborhood graph is r-regular (Chartrand and Oellermann 1993) (i.e., $|\eta(\omega)| = r$ for all $\omega \in \Omega$). For example, if $\Omega = \{\omega_1, \omega_2, \ldots, \omega_n\}$, then a neighborhood function η defined by $\eta(\omega_1) = \{\omega_2, \omega_n\}$, $\eta(n) = \{\omega_1, \omega_{n-1}\}$, and $\eta(\omega_i) = \{\omega_{i-1}, \omega_{i+1}\}$ for $i = 2,\ldots,n-1$ is 2-regular. Similarly, a neighborhood function is *connected* if the corresponding neighborhood graph is connected (Chartrand and Oellermann 1993) (i.e., for every pair of vertices $u, v \in \Omega$, there exists a path between u and v). A *tree* of size n is a connected graph having n vertices and n-1 edges (Chartrand and Oellermann 1993). A *tree neighborhood function* is a neighborhood function such that the corresponding neighborhood graph is a tree.

Note that equivalent neighborhood functions do not represent the same concept as isomorphic graphs. In particular, there can be many different neighborhood functions whose neighborhood graphs are isomorphic. On the other hand, if the neighborhood graphs of two neighborhood functions are not isomorphic, then the neighborhood functions must be distinct. This notion is explicitly stated in Lemma 1.1.

Lemma 1.1 (Armstrong and Jacobson 2008): If two neighborhood functions defined on (Ω, f) are equivalent, then their neighborhood graphs are isomorphic. On the other hand, there exist distinct neighborhood functions whose neighborhood graphs are isomorphic.

A binary vector of size $\binom{n}{2}$, where $|\Omega| = n$, can also be used to represent a neighborhood function. To see this, suppose $\Omega = \{\omega_1, \omega_2, \ldots, \omega_n\}$ such that $f(\omega_1) \leq f(\omega_2) \leq \ldots \leq f(\omega_n)$. Then there are at most $q = \binom{n}{2}$ possible edges in any graph on Ω. A vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_q) \in \{0,1\}^q$ can represent a neighborhood graph where each component of $\boldsymbol{x}$ corresponds to a different possible edge of the graph. Therefore, if $x_i = 1$ (0), then edge i is (not) in the graph. Let x_1 correspond to the edge (ω_1, ω_2), x_2 correspond to the edge (ω_1, ω_3), x_3 correspond to the edge (ω_2, ω_3), x_4 correspond to the edge (ω_1, ω_4), and so on, with x_q corresponding to the edge (ω_{n-1}, ω_n). The vector $\boldsymbol{x}$ defined in this way is called the *binary vector representation* of η. If the objective function is a one-to-one mapping, then the binary vector representation of any neighborhood function is unique. Therefore, assuming that the objective function is a one-to-one mapping, then two neighborhood functions η_1 and η_2 with binary vector representations $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively, are equivalent if and only if $\boldsymbol{x} - \boldsymbol{y} = \boldsymbol{0}$ (the zero vector in $\Re^q$). Binary vector representations of neighborhood functions are used to obtain the upcoming results.

Results are now discussed that are used to formulate expressions for the number of different neighborhood functions with $q \in Z^{+}$ edges and zero L-locals over a generic solution space, denoted by (Ω, f). To simplify the analysis, assume that all objective functions are one-to-one mappings. Lemmas 1.2 and 1.3 are needed to obtain the main results. Lemma 1.2 shows that removing the global optimum from Ω does not affect the local optimality of the remaining solutions. Lemma 1.3 states that if the search formulation (Ω, f, η) has zero L-locals, then the neighborhood function η must be connected.

Lemma 1.2 (Armstrong and Jacobson 2008): Let (Ω, f, η) be a search formulation with $|\Omega| = n$. Define $\omega^+ \in \Omega$ to be the global maximum of (Ω, f). Then, the solution $\omega \in \Omega - \{\omega^+\}$ is a L-local for (Ω, f, η) if and only if ω is a L-local for $(\Omega - \{\omega^+\}, f, \eta)$.

Lemma 1.2 implies that (Ω, f, η) and $(\Omega - \{\omega^+\}, f, \eta)$ have the same number of local minima, unless ω^+ is an isolated solution (i.e., it has no neighbors, hence its degree is zero). The global maximum ω^+ is a local minimum for a neighborhood function η if and only if it is an isolated solution in the neighborhood graph. Therefore, if the degree of ω^+ is not zero, then (Ω, f, η) and $(\Omega - \{\omega^+\}, f, \eta)$ have the same number of local minima.

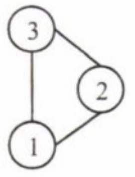
Lemma 1.3 (Armstrong and Jacobson 2008): Let (Ω, f, η) be a search formulation with $|\Omega| \geq 2$. If (Ω, f, η) has zero L-locals, then (Ω, f, η) is connected.

From Lemmas 1.2 and 1.3, a general result can be obtained on the structure of neighborhood functions with zero L-locals. Let η be a neighborhood function with zero L-locals on (Ω, f), where $|\Omega| = n$. Also, let $G = (V, E)$ be the neighborhood graph of η. Define $\{\omega_i\}$ in Ω to be the unique sequence of solutions that satisfy $f(\omega_1) > f(\omega_2) > \ldots > f(\omega_n)$. Moreover, define a sequence of graphs using the following recursive relationship: let $G_1 = G$ and for $i = 1,2,\ldots,n$, $G_{i+1} = (V_{i+1}, E_{i+1})$, where $V_{i+1} = V_i - \{\omega_i\}$, and $E_{i+1} = E_i - \{(u,v)\text{: } u = \omega_i\}$. Then, Lemmas 1.2 and 1.3 show that G_i is connected for each $i = 1,2,\ldots,n$.

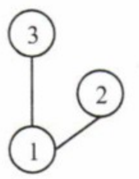
Expressions are now presented for the number of neighborhood functions with zero L-locals and the number of tree neighborhood functions with zero L-locals over a generic solution space. Lemma 1.4 gives an expression for the number of different neighborhood functions that can be defined on this solution space such that the search formulation has zero L-locals.

Lemma 1.4 (Armstrong and Jacobson 2008): Let (Ω, f) be a generic solution space with $|\Omega| = n \geq 2$. Then, the number of distinct neighborhood functions that have zero L-locals is $\prod_{j=1,2,\ldots,n-1} (2^j - 1)$.

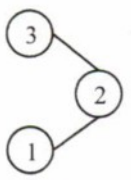
To illustrates these results, let $\Omega = \{\omega_1, \omega_2, \omega_3\}$ and $f(\omega_i) = i$ for $i = 1,2,3$. From Lemma 1.4, the total number of neighborhood functions on (Ω, f) with zero L-locals is $\prod_{j=1,2} (2^j - 1) = 1 \cdot 3 = 3$. These three neighborhood functions with zero L-locals are given below, where each circle corresponds to a solution and the objective function value is given as the label associated with each circle.



Graph 1



Graph 2



Graph 3

From Lemmas 1.2 and 1.3, all neighborhood functions with zero L-locals for the minimization problem $\Omega' = \{\omega_1, \omega_2, \omega_3, \omega_4\}$ and $f'(\omega_i) = i$ for $i = 1,2,3,4$ can be obtained by adding one or more edges to any of the three neighborhood functions given above. Thus, there are $(2^3-1)3 = 21$ neighborhood functions with zero L-locals for Ω' and f'.

Lemma 1.5 derives an expression for the total number of neighborhood functions defined on a generic solution space of size n, for any $n \geq 2$.

Lemma 1.5 (Armstrong and Jacobson 2008): Let (Ω, f) be a generic solution space with $|\Omega| = n$. Then the total number of neighborhood functions on (Ω, f) is $\prod_{i=1,2,\ldots,n-1} 2^i = 2^{\binom{n}{2}}$.

From Lemmas 1.4 and 1.5, the proportion of neighborhood functions over n solutions with zero L-locals is

$$p = \prod_{i=1,2,\ldots,n-1} (2^i - 1) \,/\, \prod_{i=1,2,\ldots,n-1} 2^i = \prod_{i=1,2,\ldots,n-1} (1 - 2^{-i}),$$

where the limit of p (as $n \to \infty$) exists (Marsden 1987) and can be approximated (numerically) to be 0.28.

Theorem 1.1 provides a closed form expression for the number of neighborhood functions with $q \in Z^+$ edges and zero L-locals on a generic solution space of size n. Theorem 1.1 is also a consequence of Lemmas 1.2 and 1.3. For $\Omega = \{1,2,\ldots,n\}$ and objective function $f(i) = i$ for $i = 1,2,\ldots,n$, define the function $\xi_n\colon Z^+ \to Z^+$ such that $\xi_n(q)$ is the number of distinct neighborhood functions η where (Ω, f, η) is connected with q edges and zero L-locals. Note that for any one-to-one mapping $g\colon \Omega \to \Re$, the number of neighborhood functions on (Ω, g) with q edges and zero L-locals is also $\xi_n(q)$.

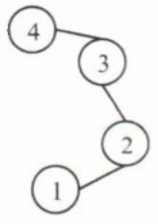
Theorem 1.1 (Armstrong and Jacobson 2008): Let (Ω, f) be a generic solution space with $|\Omega| = n$. Then, for all $n \geq 3$ and all $n - 1 \leq q \leq \binom{n}{2}$,

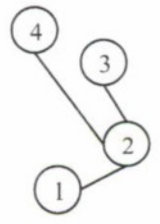
$$\xi_n(q) = (n-1)\xi_{n-1}(q-1) + \binom{n-1}{2}\xi_{n-1}(q-2) + \ldots\ldots + \binom{n-1}{q-n+2}\xi_{n-1}(n-2).$$ Furthermore, $\xi_n(q) = 0$ for all $0 \leq q \leq n\text{-}2$ and all $q \geq \binom{n}{2} + 1$.

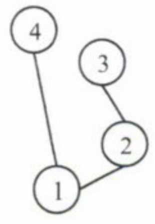
Corollary 1.1 presents an expression for the number of tree neighborhood functions with zero L-locals. Note that by Lemma 1.3, since search formulations must be connected to have zero L-locals, then the neighborhood graph for each search formulation (with zero L-locals and n-1 edges, where $|\Omega| = n$) must be a tree. Corollary 1.1 states that the number of tree neighborhood functions on Ω (with $|\Omega| = n$) with zero L-locals is equal to $(n\text{-}1)!$.

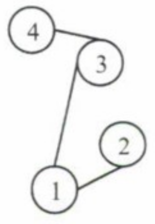
Corollary 1.1 (Armstrong and Jacobson 2008): Let (Ω, f) be a generic solution space with $|\Omega| = n$. Then $\xi_n(n\text{-}1) = (n\text{-}1)!$ for $n \geq 2$.

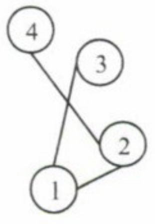
To illustrate Corollary 1.1, consider the minimization problem $\Omega = \{\omega_1, \omega_2, \omega_3\}$ and $f(\omega_i) = i$ for $i = 1,2,3$. The two neighborhood functions with zero L-locals and two edges are represented by Graphs 2 and 3 given above. Now consider the minimization problem $\Omega' = \{\omega_1, \omega_2, \omega_3, \omega_4\}$ and $f'(\omega_i) = i$ for $i = 1,2,3,4$. From Lemmas 1.2 and 1.3, all neighborhood functions with zero L-locals and three edges for this problem can be obtained by adding one edge to Graphs 2 and 3 (i.e., there are 3(2) = 6 neighborhood functions over Ω' and f' with zero L-locals and three edges). These six neighborhood functions are given below.

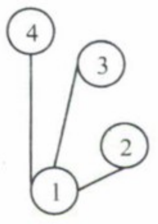

Graph 1


Graph 2


Graph 3


Graph 4


Graph 5


Graph 6

The following lemma, which is due to Cayley (see Harary and Palmer 1973), states that the number of connected labeled graphs with n vertices and n-1 edges is equal to n^{n-2}. In a labeled graph of n vertices, the integers $1,2,\ldots,n$ are uniquely assigned to each of the vertices (Harary and Palmer 1973). Therefore, a labeled graph G_l can be represented as a three-tuple $G_l = (V, E, l)$, where $G = (V, E)$ is a graph of n vertices and l: $V \rightarrow \{1,2,\ldots,n\}$ is a one-to-one mapping. Two labeled graphs $G_1 = (V_1, E_1, l_1)$ and $G_2 = (V_2, E_2, l_2)$ are the same if and only if there is a one-to-one map from V_1 to V_2 that preserves both adjacency and labeling (Harary and Palmer 1973). This means that there is a one-to-one correspondence between labeled graphs and search formulations. Lemma 1.6 states Cayley's result in terms of neighborhood functions.

Lemma 1.6 (Cayley): Let (Ω, f) be a generic solution space with $|\Omega| = n$. The number of connected neighborhood functions defined on Ω with n-1 edges is n^{n-2}.

From Corollary 1.1 and Lemma 1.6, the proportion of tree neighborhood functions with zero L-locals out of all connected neighborhood functions with n vertices and n-1 edges is $p = (n-1)! \,/\, n^{n-2}$. Corollary 1.2 contains an expression for $\xi_n(n)$ using Theorem 1.1.

Corollary 1.2 (Armstrong and Jacobson 2008): Let (Ω, f) be a generic solution space with $|\Omega| = n$. Then $\xi_n(n) = (n-1)(n-2)(n-1)!/4$ for $n \geq 3$.

Lemma 1.7 lists similar results that are obtained without using Lemmas 1.2 and 1.3.

Lemma 1.7 (Armstrong and Jacobson 2008): Let (Ω, f) be a generic solution space with $|\Omega| = n$. Then

i) for all $n \geq 2$, $\xi_n\left(\binom{n}{2}\right) = 1$,

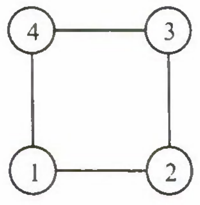
ii) for all $n \geq 3$, $\xi_n\left(\binom{n}{2} - 1\right) = \binom{n}{2} - 1$,

iii) for all $n \geq 3$, $\xi_n\left(\binom{n}{2} - 2\right) = \binom{n(n-1)/2}{2} - \binom{n}{2}$.

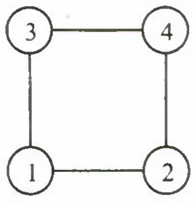
Results are presented for two classes of regular neighborhood functions. Many popular neighborhood functions are regular neighborhood functions, since neighborhood functions that are efficient and easy to implement are often regular. Examples of regular neighborhood functions used in practice are the *k*-flip neighborhood functions of Boolean optimization problems and the *k*-opt neighborhood functions of the traveling salesman problem.

Theorem 1.2 (Armstrong and Jacobson 2008): Let (Ω, f) be a generic solution space with $|\Omega| = n$. Suppose $|\Omega| = n \geq 3$. Then the number of 2-regular neighborhood functions η where (Ω, f, η) has zero *L*-locals is 2^{n-3}.

To illustrate Theorem 1.2, consider the minimization problem $\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4\}$ and $f(\omega_i) = i$ for i = 1,2,3,4. All 2-regular neighborhood functions over Ω and f with zero *L*-locals must have $\omega_2 \in \eta(\omega_1)$. The solution ω_3 can be in the neighborhood of ω_2 or ω_1. If $\omega_3 \in \eta(\omega_2)$, then $\omega_4 \in \eta(\omega_1)$ and $\omega_4 \in \eta(\omega_3)$; otherwise, if $\omega_3 \in \eta(\omega_1)$, then $\omega_4 \in \eta(\omega_2)$ and $\omega_4 \in \eta(\omega_3)$. The 2-regular neighborhood functions with zero L-locals are given below.



Graph 1



Graph 2

The number of 2-regular connected neighborhood functions on a solution space of size *n* is equal to the number of Hamiltonian cycles in a complete graph of size *n*. A *complete graph* on *n* vertices $K_n = (V, E)$ satisfies $|V| = n$ and $|E| = \binom{n}{2}$ (i.e., for all pairs of vertices $u, v \in V$, $(u,v) \in E$). For a graph $G = (V, E)$ with *n* vertices, a *walk of length k* $(k \geq 0)$, $v_0, v_1, \ldots, v_k$, is a sequence of vertices on V such that $(v_i, v_{i+1}) \in E$ for all i = 0,1,..,k-1 (see Chartrand and Oellermann 1993). A *Hamiltonian cycle* of a graph $G = (V,E)$ with $|V| = n$ vertices is defined as a walk of length n, $v_0, v_1, \ldots, v_n$, where $v_0 = v_n$ and for each vertex $v \in V$ there exists $0 \leq i \leq n$ such that $v = v_i$ (see Chartrand and Oellermann 1993). Given a discrete optimization problem (Ω, f) with $|\Omega| = n$, a Hamiltonian cycle $v_0, v_1, \ldots, v_n$ of a complete graph on Ω defines a neighborhood function η on (Ω, f) as follows: $\eta(v_0) = \{v_1, v_n\}$, $\eta(v_n) = \{v_0, v_{n-1}\}$, and $\eta(v_i) = \{v_{i-1}, v_{i+1}\}$ for $i = 1,2,\ldots,n-1$.

To count the number of Hamiltonian cycles in a complete graph K_n, note that since each vertex of V is part of every Hamiltonian cycle, then without loss of generality, fix some vertex $v^* \in V$ and represent every Hamiltonian cycle by $v^*, v_1, \ldots, v_{n-1}, v^*$. Therefore, the number of Hamiltonian cycles in K_n is equal to the number of permutations of the vertices in $V - \{v^*\}$ divided by two, since every Hamiltonian cycle $v^*, v_1, \ldots, v_{n-1}, v^*$ is equivalent to $v^*, v_{n-1}, \ldots, v_1, v^*$. Therefore, there are $(n-1)!/2$ Hamiltonian cycles in the complete graph K_n.

Lemma 1.8 (Armstrong and Jacobson 2008): Let (Ω, f) be a generic solution space with $|\Omega| = n$. Then the number of 2-regular connected neighborhood functions on (Ω, f) is $(n-1)!/2$.

From Theorem 1.2 and Lemma 1.8, the proportion of 2-regular neighborhood functions having zero *L*-locals is

$$p = 2^{n-3} / [(n-1)!/2],$$

which converges to zero (as *n* goes to infinity). Intuitively, the proportion of neighborhood functions having zero *L*-locals will converge to zero when restricted to *r*-regular neighborhood functions, where *r* is a constant. If *r* is allowed to increase as *n* increases, then the proportion of neighborhood functions having zero *L*-locals may converge to a number greater than zero.

Consider the graph that has an even number of vertices *n* and is *n*/2-regular. Property *A* defines a restricted subclass of neighborhood functions.

Property *A*: Let η be a neighborhood function on a generic solution space (Ω, f). The following property holds for η: for all $\omega \in \Omega$, if $\omega_1, \omega_2 \in \eta(\omega)$, then $\omega_1 \notin \eta(\omega_2)$.

Lemma 1.9 (Armstrong and Jacobson 2008): Let (Ω, f) be a generic solution space with $|\Omega| = n$. Suppose that *n* is even and $r = n/2$. Then

(a) the total number of *r*-regular neighborhood functions that satisfy property *A* is $\binom{n-1}{r}$,

(b) there are $\binom{n-2}{r-1}$ *r*-regular neighborhood functions that satisfy property *A* with zero *L*-locals.

By only considering *n*/2-regular neighborhood functions that satisfy property A, the proportion of such neighborhood functions η where the search formulation (Ω, f, η) has zero *L*-locals is

$$\frac{\binom{n-2}{r-1}}{\binom{n-1}{r}} = \frac{r}{n-1} = \frac{n/2}{n-1}, \text{ where } |\Omega| = n \text{ and } r = n/2.$$

Therefore, the proportion of $n/2$-regular neighborhood functions η satisfying property A with zero L-locals is slightly greater than 0.5.

To illustrate the lemmas and theorems given above, a computer program was developed to count the number of neighborhood functions with $q \in Z^{+}$ edges and zero L-locals. Given the number of vertices n, the computer program generates the adjacency matrix of every possible graph on n vertices and then counts the number of neighborhood functions with zero L-locals and q edges, for $n\text{-}1 \leq q \leq \binom{n}{2}$. The computer program also counts the number of connected neighborhood functions with q edges, for $n\text{-}1 \leq q \leq \binom{n}{2}$. Table 1 reports the results for n = 3,4,5,6. The first column contains the number of solutions in the solution space. The second column contains the number of edges in the neighborhood graph. The third column contains the number of neighborhood functions with the corresponding number of edges and solutions. The fourth column contains the number of connected neighborhood functions for the given number of edges and solutions. Similarly, the fifth column contains the number of neighborhood functions with zero L-locals and the corresponding number of edges and solutions. For example, from the first row; there are three neighborhood functions on three solutions with two edges, three connected neighborhood functions on three solutions with two edges, and two neighborhood functions on three solutions with two edges and zero L-locals.

The values reported in Table 1.1 illustrate the results reported above. Note that these values are consistent with Corollary 1.1, in that $\xi_3(2) = 2! = 2$, $\xi_4(3) = 3! = 6$, $\xi_5(4) = 4! = 24$, and $\xi_6(5) = 5! = 120$. Moreover, from Corollary 1.2, $\xi_3(3) = 2 \cdot 1 \cdot 2/4 = 1$, $\xi_4(4) = 3 \cdot 2 \cdot 6/4 = 9$, $\xi_5(5) = 4 \cdot 3 \cdot 24/4 = 72$, and $\xi_6(6) = 5 \cdot 4 \cdot 120/4 = 600$. Applying the result of Theorem 1.1 for $n = 4$ and $q = 5$, it follows that $\xi_4(5) = 3\xi_3(4) + 3\xi_3(3) + \xi_3(2) = 0 + 3 + 2 = 5$.

Table 1.1
Solution Space Structures

Number of Solutions	Number of Edges	Number of Neighborhood Functions	Number of Connected Neighborhood Functions	Number of Neighborhood Functions with Zero L-locals
n=3	2	3	3	2
	3	1	1	1
n=4	3	20	16	6
	4	15	15	9
	5	6	6	5
	6	1	1	1
n=5	4	210	125	24
	5	252	222	72
	6	210	205	98
	7	120	120	76
	8	45	45	35
	9	10	10	9
	10	1	1	1
n=6	5	3003	1296	120
	6	5005	3660	600
	7	6435	5700	1450
	8	6435	6165	2200
	9	5005	4945	2299
	10	3003	2997	1717
	11	1365	1365	923
	12	455	455	351
	13	105	105	90
	14	15	15	14
	15	1	1	1

In conclusion, the results reported provide a first step in examining the number of neighborhood functions with zero L-locals over a generic solution space. Expressions are derived for neighborhood functions that belong to certain classes, e.g., 2-regular and tree neighborhood functions. A closed form expression is derived for the number of neighborhood functions with zero L-locals that have exactly q edges, for positive integer q (Theorem 1.1). Unfortunately, this expression may take exponentially many operations (in the size of the solution space) to compute. This expression can be used, as shown in Corollaries 1.1 and 1.2, to obtain direct formulae in particular cases. The hope is that results of this type can be obtained for particular discrete optimization problems. It is highly unlikely that the methods used here to determine, for example, the number of neighborhood functions that results in zero L-Locals, can be applied for a particular discrete optimization problem. However, results of this type would be useful in establishing the limitations of local search algorithms in solving such discrete optimization problems. Given that such results would need to exploit the particular characteristics of each problem under study, problem specific techniques would need to be used, rather than the more general approach taken here over a generic solution space, though these methods may provide the necessary motivation for such research.

There are several possible extensions of this research. It would be of great value to use the results of the type provided here to develop guidelines for local search algorithms that change neighborhood functions as the algorithm executes and to develop information that can be used to select the set of a neighborhood functions to consider. Recent results with hyper-heuristic suggest that such heuristics would be particularly benefited by such insights (Burke et al. 2003). It would also be desirable to find closed form expressions or recursions for the number of practical neighborhood functions having zero L-locals, or any specific number of L-locals. If such expressions or recursions are obtained, then the expected number of L-locals can be computed for many classes of neighborhood functions. Such results may help practitioners develop local search algorithms in which the neighborhood function changes as the algorithm progress, an area of current research and investigation. Moreover, if computationally feasible expressions cannot be found, then tight upper and/or lower bounds would also be of value.

2. Using Markov Chains to Analyze the Effectiveness of Local Search Algorithms

Discrete optimization problems are defined by a large, finite set of solutions and an objective function that assigns a value to each solution. The goal when addressing a discrete optimization problem is to find solutions that globally optimize the objective function value. For NP-hard discrete optimization problems, it is unlikely that polynomial time algorithms exist to solve them (unless $P = NP$) (Garey and Johnson 1979). Moreover, complete enumeration of the entire set of solutions for large discrete optimization problem instances is typically not possible with existing computing technology. Therefore, much effort has been directed towards developing efficient heuristic algorithms to find near-optimal solutions in a reasonable amount of computing time.

For NP-hard discrete optimization problems, numerous heuristics exist for effciently finding near-optimal solutions. Local search algorithms such as simulated annealing (Kirkpatrick et al. 1983), tabu search (Glover and Laguna 1997) and threshold accepting (Dueck and Scheuer 1990) offer general approaches to finding reasonable solutions to a wide variety of NP-hard discrete optimization problems. The objective of these algorithms is to find the best possible solution using a limited amount of computing resources (see Aarts and Lentra 1997 for an in-depth discussion of local search algorithms). A further challenge is to construct algorithms that find near-optimal solutions for all instances of a particular problem, since the effectiveness of many algorithms tends to be problem-specific, as they exploit particular characteristics of problem instances (e.g., Lin and Kernighan 1973 for the traveling salesman problem). It is therefore useful to assess the performance of algorithms and devise strategies to improve their effectiveness in solving NP-hard discrete optimization problems.

The current literature on asymptotic convergence properties and finite-time performance measures focuses primarily on convergence to a globally optimal solution. However, in practice, solutions with objective function values that are close to the objective function value of a globally optimal solution are often acceptable. Without loss of generality, unless otherwise noted, assume that all discrete optimization problems are minimization problems. Orosz and Jacobson (2002) define solutions that have objective function value no greater than some threshold value as *β-acceptable solutions*, where β denotes the maximum acceptable objective function value (necessarily greater than or equal to the objective function value of a globally optimal solution for a minimization problem). Jacobson et al. (2010) analyze the finite-time behavior of local search algorithms in visiting β-acceptable solutions. They address the question: For a given GHC algorithm, what is a reasonable amount of time to search for suboptimal solutions? However, their results are limited to algorithms where independence is induced between sets of algorithm iterations, with the number of iterations in a set typically fixed at some small value. This limits the scope of local search algorithms that can be analyzed within this framework, since many local search algorithms are designed to exploit best-to-date solution information at each iteration when searching for improved solutions.

Nikolaev and Jacobson (2009) extend the results reported by Jacobson et al. (2010) to algorithms that use the best-to-date solution at each iteration (i.e., determine conditions under which convergence to β-acceptable solution occurs, and to obtain a methodology to estimate the expected number of iterations to visit a β-acceptable solution). The computational results reported focus on the analysis of the Lin-Kernighan-Helsgaun algorithm (Helsgaun 2000), which uses a variable, λ - *Opt* neighborhood search and is considered one of the most effective heuristics for solving large traveling salesman problem (TSP) instances. Nikolaev and Jacobson (2009) provide an overview of β-acceptable solutions, present a Markov chain analysis framework for local search algorithms that use information on the best-to-date solution, presents a method for computing the expected number of iterations for a local search algorithm to visit a β-acceptable solution, and introduces Markov chain pooling transformations. They also show how Markov chain pooling can be used for local search algorithm analysis and illustrate the theoretical results reported with extensive computational results for the Lin-Kernighan-Helsgaun algorithm applied to several TSP instances taken from TSPLIB (with known globally optimal values).

Local search algorithms seek to find good solutions for NP-hard discrete optimization problems by visiting inferior solutions enroute to optimal/near-optimal solutions. For a discrete optimization problem, the solution space Ω is a finite set of feasible solutions. An objective function f: $\Omega \rightarrow [0,+\infty)$ assigns a real value to each element of Ω. A neighborhood function η: $\Omega \rightarrow 2^{\Omega}$, where $\eta(\omega) \subseteq \Omega$ for all $\omega \in \Omega$, provides connections between the elements in $\omega) \subseteq \Omega$, and hence, allows the solution space to be traversed or searched by moving between solutions.

The solution space for a discrete optimization problem can be partitioned into two mutually exclusive and exhaustive sets:

- the set of globally optimal solutions, $G \equiv \{\omega^* \in \Omega: f(\omega^*) \leq f(\omega) \text{ for all } \omega \in \Omega\}$,
- the set of all other solutions, $G^c \equiv \{\omega \in \Omega: f(\omega^*) < f(\omega) \text{ for all } \omega^* \in \Omega\} = \Omega \backslash G$.

Finding a globally optimal solution for an NP-hard discrete optimization problem is often computational intensive (and may not even be possible in a reasonable amount of computing time.) Therefore, solutions that are within a predetermined threshold are often acceptable in practice. To describe such solutions, define the set of β-acceptable solutions, $D_\beta \equiv \{\omega \in \Omega: f(\omega) \leq \beta\}$, where $\beta \geq f(\omega^*), \omega^* \in G$. Note that if $\beta < f(\omega^*)$, $\omega^* \in G$, then $D_\beta = \varnothing$. Moreover, $\lim_{\beta \rightarrow f(\omega^*}\, D_\beta = G$.

Each local search algorithm run generates a sequence (random sample) of solutions. In practice, the best solution visited over the entire set of algorithm runs, not just the final solution, is reported. This allows the algorithm to aggressively traverse the solution space, visiting inferior solutions en route to a β-acceptable solution, while retaining the best-to-date solution visited. Without loss of generality, assume that all algorithm runs are initialized (stochastically) at a solution not in D_β.

Consider a local search algorithm run applied to an instance of an NP-hard discrete optimization problem. At iteration t=1,2,..., the algorithm generates a solution, the random variable ω^t. Define the best solution found over the first t iterations,

$\omega_{best}(t) \equiv \{\omega \in \{\omega^1, \omega^2, \ldots, \omega^t\}, f(\omega) \leq f(\omega^j) \text{ for all } j=1,2,\ldots,t$,

and the objective function value of the best solution found over the first t iterations $v_t \equiv f(\omega_{best}(t))$. Using these random variables, define the events

$D(t, \beta) = \{(\omega^1, \omega^2, \ldots, \omega^t): v_t \leq \beta\} \equiv$ {At least one element of D_β is visited over the first t iterations},

$D(\beta) = \{(\omega^1, \omega^2, \ldots): v_j \leq \beta \text{ for some } j = 1,2,\ldots\} \equiv$ {At least one element of D_β is visited.

Without loss of generality, assume that $P(D^c(t, \beta)) > 0$ for all t = 1,2,... (i.e., finite-time convergence to a β-acceptable solution cannot be guaranteed with probability one). The definition of $D(t, \beta)$ implies that $D^c(t-1,\beta) \supseteq D^c(t,\beta)$. Therefore, $D^c(t,\beta)$ is a telescoping, nonincreasing sequence of events in t, and hence, by the Monotone Convergence Theorem (Billingsley 1979), $P(D^c(t,\beta) \rightarrow P(D^c(\beta)$ as $t \rightarrow +\infty$, where $D^c(\beta) = \cap_{t=1,2,\ldots,+\infty} D^c(t,\beta)$.

To establish the relationship between the asymptotic convergence of a local search algorithm and the event $D(\beta)$, the following definition is needed.

Definition 2.1: *A local search algorithm converges in probability to D_β if $P(C(t,\beta)) \rightarrow 1$ as $t \rightarrow +\infty$, where $C(t,\beta) \equiv \{\omega^t \in \Omega, f(\omega^t) \leq \beta\} \equiv$ {An element of D_β is visited at iteration t}.*

Given an initial solution $\omega^0 \in \Omega$, if a local search algorithm converges in probability to D_β (as $t \rightarrow +\infty$), then $P(D(\beta))=1$. Equivalently, if $P(D(\beta)) < 1$, then the algorithm does not converge in probability to D_β (i.e., for all $\varepsilon > 0$, there exists some iteration $t_0(\varepsilon)$ such that $P(C(t,\beta)) \leq 1 - \varepsilon$ for all $t \geq t_0(\varepsilon)$.

The β-acceptable solution problem asks whether a local search algorithm will eventually visit an element of D_β, given that the algorithm, after executing a finite number of iterations, has yet to visit an element of D_β.

The random variable $\tau_\beta = \min\{t \leq l: v_t \leq \beta\}$ measures the number of iterations needed for a local search algorithm to visit an element of D_β for the first time, and hence, provides a measure for its effectiveness.

A Markov chain framework is now described that will be used to obtain performance measures for local search algorithms. Consider a local search algorithm guided by information on the best-to-date solution at each iteration. The conditional probability that the objective function value of a solution at iteration t+1 is less than or equal to some value
$x \in (0, +\infty)$, given the best solution found through the first t iterations, is denoted $P(f(\omega^{t+1}) \leq x \mid \omega_{best}(t))$; refer to this probability as the conditional cumulative distribution function (or simply, conditional CDF) of the objective function value of a solution at iteration t+1.

Consider an ordered set of l β-values $\{\beta_i: i=0,1,2,...,l-1\}$ such that $f(\omega^*)=\beta_0 < \beta_1 < \beta_2 < ... < \beta_{l-1} < +\infty$. Define a set of intervals $\{\Delta_i : i = 0,1,2,...,l\}$ such that $\Delta_0 = [\beta_0, \beta_0]$, $\Delta_i = [\beta_{i-1}, \beta_i]$, for i = 1,2,..., l-1, and $\Delta_l = [\beta_{l-1}, +\infty]$. By definition, these l+1 intervals $\{\Delta_i : i = 0,1,2,...,l\}$ are non-overlapping, with $\cup_{i=0,1,...,l} \Delta_i = [f(\omega^*),+\infty)$. Also, for all i = 0,1,2,...,l, $f(\omega) \in \cup_{j=0,1,...,i} \Delta_j$, for all $\omega \in D_{\beta_i}$.

First, restrict the analysis to a problem instance where each feasible solution has a unique objective function value. Since the set of feasible solutions is finite, then all solutions can be ordered and indexed such that $f(\omega^*) < f(\omega_1) < f(\omega_2) < ... < f((\omega_l)$, where $l = |\Omega|$, $\beta_0 = f(\omega^*)$ and $\beta_i = f(\omega_i)$ for i=1,2,...,l. Then, each interval Δ_i, i=0,1,2,...,l, contains exactly one feasible solution.

Consider the stochastic process $\{X_t\}$, t=0,1,2,..., the set of interval indices generated by successive iterations of a local search algorithm. For each iteration t=0,1,..., let $X_t = i$ if and only if $v_t \in \Delta_i$, i=0,1,...,l. Theorem 2.1 shows that this stochastic process has the Markov property, and hence, $\{X_t\}$ is a Markov chain.

Theorem 2.1 (Nikolaev and Jacobson 2009): *$\{X_t\}$ has the Markov property.*

The key relationship needed to obtain Theorem 2.1 is $\{v_t \in \Delta_{X_t}\} \Leftrightarrow \{v_t = f(\omega_{X_t})$, which is only true when each interval $\{\Delta_i : i=0,1,..,l\}$ contains exactly one feasible solution. To obtain an expression for the expected number of iterations to visit a β-acceptable solution, for some $\beta \geq f(\omega^*)$, construct a transition matrix H for $\{X_t\}$, t = 0,1,..., with states i=0,1,...,l. Let states $\{0,1,...,i_\beta\}$ be absorbing states, where $i_\beta = \{i : \beta \in \Delta_i\}$, and $H_{i,j} = P(v_{t+1} \in \Delta_j \mid v_t \in \Delta_i) = P(f(\omega^{t+1}) = f(\omega_j) \mid \omega_{best}(t) = \omega_i)$ for $i = i_\beta+1, i_\beta+2, ..., l$ and j = 0,1,...,l.

Lemma 2.1 establishes a relationship between the expected time to visit an element of D_β and the matrix H.

Lemma 2.1 (Nikolaev and Jacobson 2009): *$E(\tau_\beta)$ is the expected time to absorption for the Markov chain $\{X_t\}$ with transition matrix H.*

Lemma 2.1 shows that the value of $E(\tau_\beta)$ can be computed using the transition matrix H. However, this result has no practical value, since the dimension of H (namely $|\Omega|$) is very large for any reasonable problem instances. In practice, $|\Omega|$ grows exponentially with the dimension of the problem, which makes it impractical to directly analyze the resulting Markov chain. To overcome this difficulty, a Markov chain with fewer states can be constructed. To this end, consider the non-overlapping intervals $\{\Delta'_u : u = 0,1,..,U\}$, with $U << |\Omega|$ (i.e., several solutions fall into each interval). Therefore, all solutions falling into a single interval can only be analyzed as a group, which also means that the assumption that each solution has a distinct objective function value must be relaxed (i.e., several different solutions may have the same objective function value). One issue that immediately arises when treating this more general case is that the stochastic process $\{X_t\}$, t=0,1,..., is no longer Markovian. This suggests that a new Markov chain must be created for computing the expected time to visit D_β.

Given a Markov chain with a large number of states (termed the *original* Markov chain, such as described above, it is possible to construct a new Markov chain with fewer states, such that its expected time to absorption is equal to the expected time to absorption for the original Markov chain. Theorem 2.2 describes how such a construction can be achieved for a Markov chain with a lower-triangular transition matrix.

Theorem 2.2 (Nikolaev and Jacobson 2009): *Given any N state absorbing Markov chain $\{Y_t\}$ with lower-triangular transition matrix T, where state one is absorbing, and $P(Y_0=n) = P_n$ for n=1,2,...,N is the distribution for the initial state Y_0, construct an $(N-1) \times (N-1)$ transition matrix T', such that*

a) $T'_{i,j} = T_{i,j}$ for i = 1,2,...,N-2, j = 1,2,...,N-1,

b) $T'_{N-1,j} = (\omega_{N-1} / (\omega_{N-1} + \omega_N))T_{N-1,j} + (\omega_N / (\omega_{N-1} + \omega_N))T_{N,j}$ for j = 1,2,...,N-2,

c) $T'_{N-1,N-1} = 1 - \Sigma_{j=1,2,...,N-2} T'_{N-1,j}$,

where ω_{N-1} and ω_N are the expected number of visits to states N-1 and N, respectively, prior to absorption in the Markov chain $\{Y_t\}$. Then, the expected time to absorption for the N-1 state Markov chain $\{Y'_t\}$ with transition matrix T', where state one is absorbing, and $P(Y'_0=n) = P'_n = P_n$ for n=1,2,...,N-2, and $P(Y'_0=N-1) = P'_{N-1}=P_{N-1}+P_N$ is the distribution for the initial state Y'_0, is equal to the expected time to absorption for the Markov chain $\{Y_t\}$.

Theorem 2.2 shows that for any absorbing Markov chain with a lower-triangular transition matrix and a given initial state distribution, the last two states in the chain can be pooled into a single state to form a new transition matrix, such that the expected time to absorption for the Markov chain defined by this new transition matrix is equal to the expected time to absorption for the original chain (the initial state distribution for the new chain is the same as the original chain, with the probabilities for the two pooled states summed). Note that the expected number of visits to the pooled state prior to absorption in the new chain, ω'_{N-1}, is equal to the sum of the expected numbers of visits to the two states that are being pooled prior to absorption in the original chain,

$$\omega'_{N-1} = P'_{N-1} / (1-T'_{N-1,N-1}) = \omega_{N-1} + \omega_N.$$

Corollary 2.1 follows from Theorem 2.2, by considering more than two states being pooled.

Corollary 2.1 (Nikolaev and Jacobson 2009): *Given any N state absorbing Markov chain $\{Y_t\}$ with lower-triangular transition matrix T, where state one is absorbing, and $P(Y_0 = n) = P_n$ for $n=1,2,\ldots,N$ is the distribution for the initial state Y_0, construct an $(N-m+1) \times (N-m+1)$, for some $m=2,3,\ldots,N-1$, transition matrix T', such that*

a) $T'_{i,j} = T_{i,j}$ for $i=1,2,\ldots,N-m$, $j=1,2,\ldots,N-m+1$,

b) $T'_{N-m+1,j} = \Sigma_{j=N-m,N-m+1,\ldots,N}\ \omega_i T'_{i,j} / (\Sigma_{j=N-m,N-m+1,\ldots,N}\ \omega_i T'_{i,j})$ for $j=1,2,\ldots,N-m+1$,

c) $T'_{N-m+1,N-m+1} = 1 - \Sigma_{j=1,2,\ldots,N-m}\ T'_{N-m+1,j}$,

where ω_i is the expected number of visits to state $i=N-m+1,N-m+2,\ldots,N$, prior to absorption for the Markov chain $\{Y_t\}$. Then, the expected time to absorption for the N-1 state Markov chain $\{Y'_t\}$, with transition matrix T', where state one is absorbing, and $P(Y'_0 = n) = P'_n = P_n$ for $n=1,2,\ldots,N-m-1$, and $P(Y'_0 = N-m) = P'_{N-m} = \Sigma_{j=N-m,N-m+1,\ldots,N}\ P_i$ is the distribution for the initial state Y'_0, is equal to the expected time to absorption for the Markov chain $\{Y_t\}$.

Corollary 2.1 shows that for any absorbing Markov chain with a lower-triangular transition matrix and a given distribution of the initial state, the last m (that can take on a value between two and the total number of states in the chain minus one) states in the chain can be pooled into a single state, resulting in a new transition matrix, such that the expected time to absorption in a Markov chain defined by this new transition matrix is equal to the expected time to absorption in the original chain (the initial state distribution for a new chain is the same as the original, with the probabilities for the pooled states summed). However, it does not provide results for the case where the pooled states are not the last states of the chain. Lemma 2.2 is needed to address this case.

Lemma 2.2 (Nikolaev and Jacobson 2009): *Given any N state absorbing Markov chain $\{Y_t\}$ with lower-triangular transition matrix T, where state one is absorbing, and $P(s(0)=n)=P_n$ for $n=1,2,\ldots,N$ is the distribution for the initial state s(0), construct an $(N-1) \times (N-1)$ transition matrix T' by removing the last column and the last row in T. Then, μ, the expected time to absorption for the Markov chain $\{Y_t\}$, is equal to $P_N / (1-T_{N,N})$ plus μ', the expected time to absorption for the N-1 state Markov chain $\{Y'_t\}$, with transition matrix T', where state one is absorbing, and $P(s'(0)=n) = P'_n = P_n + T_{N,n} P_N / (1-T_{N,N})$ for $n=1,2,\ldots,N-1$, is the distribution for the initial state s'(0).*

Lemma 2.2 gives an expression for the expected time to absorption for Markov chains with lower-triangular transition matrices. Theorem 2.3 provides a general result for computing the expected times to absorption for such Markov chains, using state pooling. It establishes that any consecutive states in an absorbing Markov chain with a lower-triangular transition matrix can be pooled, such that the result in Corollary 2.1 holds.

Theorem 2.3 (Nikolaev and Jacobson 2009): *Given any N+r state absorbing Markov chain $\{Y_t\}$ with lower-triangular transition matrix T ($r \in Z^+$), where state one is absorbing, and $P(Y_0=n) = P_n$ for $n=1,2,\ldots,N+r$ is the distribution for the initial state Y_0, construct an $(N-m+r+1) \times (N-m+r+1)$ transition matrix T', for some $m=2,3,\ldots,N-1$, such that*

a) $T'_{i,j} = T_{i,j}$ for $i=1,2,\ldots,N-m$, $j=1,2,\ldots,N-m+r+1$,

b) $T'_{N-m+1,j} = \Sigma_{j=N-m,N-m+1,\ldots,N}\ \omega_i T'_{i,j} / (\Sigma_{j=N-m,N-m+1,\ldots,N}\ \omega_i T'_{i,j})$ for $j=1,2,\ldots,N-m+1$,

c) $T'_{N-m+1,N-m+1} = 1 - \Sigma_{j=1,2,\ldots,N-m}\ T'_{N-m+1,j}$,

d) $T'_{N-m+1,j} = 0$ for $j = N-m+2, N-m+3,\ldots,N-m+r-1$,

e) $T'_{i,j} = T_{i+m-1,j}$ for $i= N-m+2, N-m+3,\ldots,N-m+r-1$, $j = 1,2,\ldots,N-m$,

f) $T'_{i,N-m+1} = \Sigma_{j=N-m+1,N-m+2,\ldots,N}\ T_{i+m-1,j}$ for $1 = N-m+2, N-m+3, \ldots, N-m+r-1$,

g) $T'_{i,j} = T_{i+m-1,j}$ for $i= N-m+2, N-m+3,\ldots,N-m+r-1$, $j = N-m+2, N-m+3,\ldots,N-m+r-1$,

where ω_i is the expected number of visits to state $i= N-m+1,N-m+2,\ldots,N$, prior to absorption for the Markov chain $\{Y_t\}$. Then, the expected time to absorption for the N-m+r+1 state Markov chain $\{Y'_t\}$, with transition matrix T', where state one is absorbing, and $P(Y'_0=n) = P'_n = P_n$ for $n=1,2,\ldots,N-m$, $P(Y'_0=N-m+1)=P'_{N-m+1} = \Sigma_{j=N-m,N-m+1,\ldots,N}\ P_i$, and $P(Y'_0=n) = P'_n=P_{n+m-1}\}\$$ for $n = N-m+2,N-m+3,\ldots,N-m+r+1$ is the distribution for the initial state Y'_0, is equal to the expected time to absorption for the Markov chain $\{Y_t\}$.

To see how Theorem 2.3 can be used to obtain computational results for local search algorithms, assume that each execution of a local search algorithm is initialized at a randomly generated initial solution ω_0, and hence, $P(f(\omega^1) \leq x \mid \omega_0)$ is the conditional CDF of the objective function value of a solution from iteration one. For a problem instance where each distinct solution has a unique objective function value, the stochastic process $\{X_t\}$, t=1,2,..., is a Markov chain with transition matrix H. Using H, the expected time to absorption in $\{X_t\}$, $E(\tau_\beta)$, can be computed. Without loss of generality, let B denote a set of β-values for the objective function, with U=|B|, and select U+1 intervals $\{\Delta'_u: u = 0,1,\ldots,U\}$ such that $\cup_{u=0,1,\ldots,U} \Delta'_u = \cup_{i=0,1,\ldots,I} \Delta_i$, $\Delta'_u = \cup_{\{i:\, f(\omega_i) \in \Delta_u\}} \Delta_i$ for all u = 0,1,...,U, and each β∈B is the largest value in the interval Δ'_{u_β}, where $u_\beta = \{u: \beta \in \Delta'_u\}$ for β∈ B. By Theorem 2.3, construct a new transition matrix H' from H by pooling each group of states, for $v_t \in \Delta'_u$, into a single state, for u = 0,1,...,U. Without loss of generality, assume that the initial state distribution for the Markov chain $\{X_t\}$ is given as a probability mass function. Theorem 2.3 establishes that the expected time to absorption in the Markov chain with transition matrix H' (the initial state distribution for this chain is the same as the initial state distribution for $\{X_t\}$, with the probabilities for the pooled states summed) is equal to $E(\tau_\beta)$, the expected time to absorption in $\{X_t\}$.

Note that in practice, local search algorithms can not be executed for an infinite number of iterations. However, they are often executed with multiple restarts, with an upper bound T set on the number of iterations an algorithm performs during each restart run. Given T, let $E_T(\tau_\beta)$ denote the expected number of iterations to visit D_β. Since τ_β is a non-negative random variable, then

$$E_T(\tau_\beta) = \Sigma_{t=0,1,\ldots} P(\tau_\beta > t) = 1+\Sigma_{t=12,\ldots} P(\tau_\beta > t)$$

For an algorithm executed with multiple restarts, the runs (at most T iterations each) are independent of each other. For any run, the probability that D_β is not visited during the first t iterations, F(t), is the same. Therefore, for any t=1,2,...,

$$P(\tau_\beta > t) = F(T)^{\lfloor t/T \rfloor} F(t-T\lfloor t/T \rfloor),$$

and hence,

$$E_T(\tau_\beta) = 1+\Sigma_{t=1,2,\ldots,T} P(\tau_\beta > t) +\Sigma_{t=T+1,T+2,\ldots,2T} P(\tau_\beta > t) + +\Sigma_{t=2T+1,2T+2,\ldots,3T} P(\tau_\beta > t) + \ldots$$
$$=1+\Sigma_{t=1,2,\ldots,T} F(t) +F(T)\, \Sigma_{t=1,2,\ldots,T} F(t) + F(T)^2\, \Sigma_{t=1,2,\ldots,T} F(t) + F(T)^3\, \Sigma_{t=1,2,\ldots,T} F(t) \ldots$$
$$= 1 + ((\Sigma_{t=1,2,\ldots,T} F(t)) / (1 - F(T)))$$

which holds only if F(T) < 1. Otherwise, the probability that D_β is visited during any finite number of independent algorithm runs (with each at most T iterations) is equal to zero, and hence, $E((\tau_\beta) = +\infty$.

Let states i = 0,1,...,i_β in the Markov chain $\{X_t\}$ with transition matrix H be absorbing states. For any t=1,2,...,T, F(t) can be computed using the matrix $H^{(t)}$, the tth power of H, and the initial state distribution for the Markov chain $\{X_t\}$,

$$F(t) = 1-\Sigma_{i=i\beta+1,\ldots,I}\, \Sigma_{j=01,2,\ldots,i\beta}\, H^{(t)}_{i,j}\, P(\omega^0 \in \Delta_i).$$

Using matrix $H'^{(t)}$, the tth power of transition matrix H', and the initial state distribution for the Markov chain with transition matrix H', for any t=1,2,..,T, define

$$F'(t) \equiv 1-\Sigma_{u=u\beta+1,\ldots,U}\, \Sigma_{v=01,2,\ldots,u\beta}\, H'^{(t)}_{u,v}\, P(\omega^0 \in \Delta'_u)$$

and

$$E'_T \equiv 1 + ((\Sigma_{t=1,2,\ldots,T} F'(t)) / (1 - F'(T)))$$

In general, F'(t) ≠ F(t) for any t=1,2,..., and hence, $E'_T \neq E_T(\tau_\beta)$ for any fixed T=1,2,.... Theorem 2.4 establishes a convergence result that allows the transition matrix H' to be used to estimate the expected number of iterations that a local search algorithm requires to visit D_β when executed with multiple restarts.

Theorem 2.4 (Nikolaev and Jacobson 2009): *For all $\varepsilon > 0$, there exists $T^* \in I$ such that $|E'_T - E_T(\tau_\beta)| < \varepsilon$ for all $T \geq T^*$.*

Nikolaev and Jacobson (2009) reports theoretical and computational results for estimating the expected number of iterations to visit D_β based on the Markov chain with transition matrix H'. To illustrate the estimation procedure, the Lin-Kernighan-Helsgaun (LKH) algorithm was applied to eight medium and large TSP instances taken from TSPLIB (PCB442, PR1002, RL1889, D2103, U2152, PR2392, PCB3038 and FNL4461, with the number denoting the number of cities; see Reinelt 1991) to obtain estimates for $E_T(\tau_\beta)$ for each instance. The Lin-Kernighan algorithm (Lin and Kernighan 1973) uses variable λ-Opt neighborhoods, where at each inner loop iteration, the algorithm considers a growing set of λ-Opt moves (starting with λ=2); see Aarts and Lenstra (1997) for a detailed description. Helsgaun (2000) extends the work of Lin and Kernighan to describe the LKH algorithm, a highly effective heuristic for obtaining near-optimal solutions for large TSP instances. Note that these experiments were not designed to present a new local search heuristic for the TSP, but rather, to demonstrate a method to analyze local search algorithms applied to the TSP (or in fact, any hard discrete optimization problem).

A description of the design of such experiments, including convergence results for the resulting estimators, is now given. Consider K independent LKH runs. For any $u \in Z^+$, $u_\beta < u \leq U$, define $S(k,T,\Delta'_u)$ as the set of all iterations (among the first T iterations of the kth run) such that the objective function value of each such iteration falls into the interval Δ'_u, with $c(k,T, \Delta'_u) \equiv |S(k,T, \Delta'_u)|$. For any $v \in Z$, $0 < v \leq u$, define

$\tilde{H}'_{u,v} = \Sigma_{k=1,2,\ldots,K} \Sigma_{s\in S(k,T,\Delta'u)} I_s / \Sigma_{k=1,2,\ldots,K}\, c(k,T,\Delta'u)$

where $I_s = 1$ if a solution found at iteration $s \in S(k,T, \Delta'u)$, $k=1,2,\ldots,K$, belongs to $\Delta'v$, and 0 otherwise. Theorem 2.5 establishes a convergence result for $\tilde{H}'_{u,v}$.

Theorem 2.5 (Nikolaev and Jacobson 2009): *For those values of β such that $P(D_\beta) > 0$, $\tilde{H}'_{u,v} \rightarrow_P H'_{u,v}$ as $T \rightarrow +\infty$ and $K\rightarrow +\infty$ for any $u,v \in Z^+$ with $u_\beta < u \leq U$ and $0<v \leq u$.*

Theorem 2.5 establishes that the given estimation procedure guarantees weak convergence of $\tilde{H}'_{u,v}$ to $H'_{u,v}$ as the number of runs and the number of iterations in each run grow infinitely large. Define the random variables

$\tilde{F}'(t) \equiv 1 - \Sigma_{u=u\beta+1,\ldots,U}\ \Sigma_{v=0,\ldots,u\beta}\ \tilde{H}'^{(t)}_{u,v}\ P(\omega^0 \in \Delta'_u)$,

and

$\tilde{E}'_T \equiv 1 + ((\Sigma_{t=1,2,\ldots,T}\ \tilde{F}'(t)\,) / (1 - \tilde{F}'(T)\,))$

Theorem 2.6 establishes that $\tilde{E}'_T$ converges in probability to E'_T as $T \rightarrow +\infty$ and $K \rightarrow +\infty$.

Theorem 2.6 (Nikolaev and Jacobson 2009): *For those values of β such that $P(D_\beta) > 0$, $\tilde{E}'_T \rightarrow_P E'_T$ as $T \rightarrow +\infty$ and $K\rightarrow +\infty$*

Nikolaev and Jacobson (2009) report computational results to illustrate how the estimator for the expected number of iterations to visit D_β can be computed for a local search algorithm with multiple restarts. First, the definitions and a general description of the LKH are given. Using the terminology introduced by Helsgaun (2000), each iteration of the LKH is termed a *trial*. The LKH algorithm begins a trial by randomly generating an initial solution, which is iteratively improved using the variable λ-Opt neighborhood function. A trial ends when a local minimum is attained. Define a *run* as a set of trials. The maximum number of trials in a run, T, is a user-defined input parameter, which is typically the number of cities for a given TSP problem. However, if the globally optimal solution value is specified as one of the LKH input parameters, a run may end prematurely, in which case a trial returns a globally optimal solution. A *replication* is a set of K runs. The experimental data from a single replication is used to estimate H', which in turn is used to compute $\tilde{E}'_T$.

Thirty replications of K=100 independently seeded LKH runs were executed to estimate $E_T(\tau_\beta)$, using $\tilde{E}'_T$. The maximum number of trials in a run, T, was set equal to the number of cities for each given TSP instance. The resulting data was then used to compute $\tilde{E}'_T$ (and the associated sample standard deviation estimator $s_{E'T}$ for each value of β. All these values are reported in Table 2.1.

To assess the validity of these estimates for $E_T(\tau_\beta)$, the LKH was modified such that a set of runs were executed until $R_V = 500$ independently seeded replications visited D_β, where each run was reinitialized by resetting the algorithm with a new randomly generated initial solution. The resulting data was then used to compute the mean and sample standard deviation estimates for τ_β; these values are also reported in Table 2.1. To statistically compare the estimators, two-sided hypothesis tests were performed for different values of β, with null hypothesis H_0: $E'_T - E_V[\tau_\beta] = 0$ and alternative hypothesis H_0: $E'_T - E_V[\tau_\beta] \neq 0$; the resulting test statistic values and the associated p-values are reported in Table 2.1.

The third and fourth column values reported in Table 2.1 represent the number of trials required to visit β-acceptable solutions. For example, for problem RL1889, the estimator predicts that 803 LKH trials are required (on average) to visit a solution that is within 0.01% of the optimal solution. In Table 2.1, when the p-value associated with estimator $\tilde{E}'_T$ is greater than $0.05$, then the corresponding estimated value is highlighted in bold; such values mean that the point estimator and the validation estimate for $E(\tau_\beta)$ are statistically indistinguishable (i.e., with a Type I error of $\alpha = 0.05$). Note that the LKH computer experiments were performed using the LKH-1.3 package (LKH 2005), written in C. All computer experiments were executed on a DELL OptiPlex G620, 3GHz Pentium D with 2GB of RAM.

Among the fifteen LKH algorithm results reported in Table 2.1 that were validated, all the $\tilde{E}'_T$ estimated values are statistically indistinguishable from the validation estimates $\bar{E}_T(\tau_\beta)$, at the $\alpha = .05$ level. Table 2.2 reports the computation times for the estimation and validation phases for the eight test problems. The computer experiment CPU times (per set of 100 LKH runs) for each TSP instance for the $\tilde{E}'_T$ estimation experiments ranged from between 195 CPU seconds and 20.5 CPU hours. The execution times for each completed validation experiment (all $R_V = 500$ replications) ranged from between 4723 CPU seconds and 1918 CPU hours (around 80 CPU days), based on the size of the TSP instance. The results from Table 2.2 suggest that the proposed estimation procedure is computationally efficient in obtaining good estimates of $E_T[\tau_\beta]$.

Table 2.1
Lin-Kernighan-Helsgaun Algorithm Results for Estimating $E_T(\tau_\beta)$

Problem		Statistics		Hypothesis Test	
Instance	β/f^*	$(\bar{E}'_T, \frac{s_{E'_T}}{\sqrt{30}})$	$(\bar{E}_T(\tau_\beta), \frac{s_T(\tau_\beta)}{\sqrt{500}})$	Z	p-value
PCB442	1	(107, 5)	(106, 8)	0.07	0.95
$f^* = 50778$					
PR1002	1	(305, 9.2)	(297, 12.5)	0.48	0.63
$f^* = 259045$	1.0001	(92, 5.4)	(86, 6.9)	0.67	0.50
RL1889	1	(1879, 114.8)	(1913, 106)	-0.23	0.82
$f^* = 316536$	1.0001	(803, 88)	(651, 55)	1.46	0.14
D2103	1.0001	(64745, 7734.7)	(50503, 2069.6)	1.78	0.08
$f^* = 80450$	1.0002	(7574, 301.2)	(6437, 297)	1.51	0.13
U2152	1	(7788, 273.4)	(7155, 288.37)	1.59	0.11
$f^* = 64253$	1.0001	(3220, 88.3)	(3315, 163.4)	-0.51	0.61
PR2392	1	(303, 3.9)	(300, 9.4)	0.32	0.75
$f^* = 378032$	1.0001	(86, 1.1)	(90, 3.4)	-0.9	0.37
PCB3038	1	(2152, 23.2)	(2249, 115.5)	-0.82	0.41
$f^* = 137694$	1.0001	(1506, 24.7)	(1354, 87)	1.68	0.09
FNL4461	1	(7145, 231.6)	(7490, 323.7)	-0.87	0.39
$f^* = 182566$	1.0001	(202, 3.9)	(210, 15.8)	-0.53	0.6

Table 2.2
Lin-Kernighan-Helsgaun Algorithm Computation Times

Problem Instance	*Estimation* *CPU*	*Validation* *CPU*
Pcb442 (f^* = 50778)	201 sec.	4723 sec.
Pr1002 (f^* = 259045)	195 sec.	47519 sec.
Rl1889 (f^* = 316536)	4170 sec.	684376 sec.
D2103 (f^* = 80450)	15 hr.	1793 hr.
U2152 (f^* = 64253)	18389 sec.	341218 sec.
Pr2392 (f^* = 378032)	2700 sec.	16600 sec.
Pcb3038 (f^* = 137694)	8.9 hr.	585 hr.
Fnl4461 (f^* =182566)	20.5 hr.	1918 hr.

The data collection for estimating $E(\tau_\beta)$ can be computationally intensive and in some cases, redundant. Work is in progress to develop estimation techniques to collect data as the algorithm executes, rather than collecting data off-line and applying the results to a local search algorithm retrospectively. Another limitation of the computational analysis presented here is the requirement for the local search algorithm to be executed with multiple restarts. However, this is not a major drawback, since such a setting is the most typical for practical use of local search algorithms. More practical questions can be addressed in the future if an expression for the rate of convergence of the estimator $\tilde{E}'_T$ to $E_T(\tau_\beta)$ as a function of T and K is obtained; this is an active current area of investigation.

This research is intended as a stepping stone towards developing a general framework for providing *prospective* information on local search algorithm performance. Using an array of statistical tools to analyze the conditional CDF, $P(f(\omega^{t+1} \leq x \mid \omega_{best}(t))$, as a function of the best-to-date solution $\omega_{best}(t)$, may be the key for this effort. Work is in progress to determine how these distributions evolve as the current best solution approaches global optima, and to predict values of $E_T(\tau_\beta)$ for values of β that have not yet been reached.

3. A Post-Optimality Framework for Multi-Criteria Optimization

Multi-objective optimization algorithms can generate large sets of Pareto optimal (non-dominated) solutions. Identifying the best solutions across a very large number of Pareto optimal solutions can be a challenge. Therefore it is useful for the decision-maker to be able to obtain a small set of *preferred* Pareto optimal solutions. Kao and Jacobson (2008) introduce a discrete optimization problem framework for obtaining optimal subsets of solutions from large sets of Pareto optimal solutions. Kao and Jacobson (2008) prove this problem to be *NP*-hard, and provide two exact algorithms and five heuristics to address this problem. They also reports computational results with five test problems, to compare the performances of these algorithms and heuristics. The results suggest that preferred subset of Pareto optimal solutions can be efficiently obtained using the heuristics, while for smaller problems, exact algorithms should be applied.

Many real-world optimization problems involve multiple (and often conflicting) objectives. These problems are relevant in a variety of engineering disciplines, scientific fields, and various industrial applications (Coello et al. 2002, Ehrgott and Gandibleux 2002). Unlike single objective optimization problems, where one attempts to find the best solution (global optimum), in multi-objective optimization problems, there may not exist one solution that corresponds to the best with respect to all objectives. Solving a multi-objective optimization problems consist of generating the Pareto frontier (i.e., the set of non-dominated solutions that represents the trade-off among the objective function values. Different approaches have been explored to approximate and generate such sets of Pareto optimal solutions. Some interactive approach incorporates preferences into the optimization procedure to explore a specific region of the solution space, while other approaches focus on generating a diverse set of Pareto optimal solutions. Such sets of Pareto optimal solutions can be extremely large, which motivates the need for post-optimality analysis for multi-objective optimization problems. The area of post-optimality analysis addressed by Kao and Jacobson (2008) focuses on obtaining a preferred subset of solutions from a very large set of solutions with acceptable objective function values. The goal in obtaining large sets of Pareto optimal solutions is to provide the decision-maker with a diverse set of such solutions. Although obtaining diverse Pareto optimal solutions is important, it is often impractical for a human decision-maker to manually examine each such solution, and hence, efficiently identify a good subset of such solutions. Previous research in this area has focused on generalizing the representation of the full set of Pareto optimal solutions with a smaller subset (Kasprzak and Lewis 2000, Mattson et al. 2004). Such procedures are not post-optimality analysis procedures, but rather, extensions to multi-objective optimization procedures, which are designed to generate diverse sets of Pareto optimal solutions (Messac and Mattson 2004, Messac et al. 2003, Kasprzak and Lewis 2001). Another area of research that incorporates preferences into the optimization procedures are interactive methods (Miettinen 1999, Miettinen and Makela 2000). These interactive methods provide a decision-maker with better control over the optimization process, allowing them to explore specific regions of the search space. However, solutions obtained are quite sensitive towards the preferences of the decision-maker. These approaches also require the decision-maker to have a thorough knowledge of the problem. Korhonen and Halme (1990) suggest the use of a value function in helping decision-makers to identify the most preferred solutions. Alternatively, to objectively evaluate and distinguish good subsets of Pareto optimal solutions, Das (1999) proposes an ordering and degree of efficiency among Pareto optimal solutions, which provides a way to measure and prune out less desirable Pareto optimal solutions. The general problem of post-optimality has not been addressed, in it purest form, due to his extreme difficulty and the inability to create a mathematical framework to model it.

Kao and Jacobson (2008) introduce and analyze a discrete optimization problem framework for obtaining a preferred subset of Pareto optimal solutions from a larger set. This framework alleviates the sensitivity of value function approaches, while obtaining a desired size subset of Pareto optimal solutions. They introduce two exact algorithms for solving the discrete optimization problem. They also provide five heuristics that obtain near-optimal solutions. The complexity of the discrete optimization problem formulation is presented. The exact algorithms and heuristics are applied to five test problems of various sizes, to provide comparisons of their computational performances.

Kao and Jacobson (2008) describe in detail the discrete optimization problem framework for post-optimality analysis of Pareto optimal solutions. Top describe this framework, consider the multi-objective optimization problem:

Min $F(x) = (f1(x), f2(x), \ldots, fk(x)) = z = (z1, z2, \ldots, zk)$
subject to: $x \in S$

with k (≥ 2) objective functions $fi : \Re^n \rightarrow \Re$, $i = 1, 2, \ldots, k$, where the decision variables $x = (x1, x2, \ldots, xn)$ belong to the feasible region $S \subseteq \Re^n$.

Kao and Jacobson (2008) provide formal definitions for a Pareto optimal solution, a value function, a percentile vector (of each Pareto optimal solution), and the percentile space. Let $S^{PO} = (x^1, x^2, \ldots, x^N) \subseteq S$ denote a set of Pareto optimal solutions, which may not contain the complete set of all Pareto optimal solutions. By definition, given a set of Pareto optimal solutions, S^{PO}, for every $x^j \in S^{PO}$ $j = 1, 2, \ldots, N$, there exists a unique percentile vector pj. Therefore, there is a one-to-one mapping for all $x \in S^{PO}$ to some $pj \in \wp k$, termed the *percentile set* (formally defined in Kao and Jacobson 2008). A percentile function $q : (0,1]^k \rightarrow \Re$ is a value function on the percentile space.

The defined preferred Pareto optimal solution subset(s) can be obtained by solving the following discrete optimization problem, which optimizes the percentile function q.

max $q(p1, p2, \ldots, pk)$
subject to: $|N_{sub}| \geq N'$

where $N_{sub} = \{x \in S^{PO} : pi(x) \geq pi, i = 1, 2, \ldots, k\}$, N' is the minimum number of solutions in the preferred subset of Pareto optima S^{PO}, and pi, $i = 1, 2, \ldots, k$, correspond to the percentile *threshold* for each of the k objectives. This discrete optimization problem formulation is termed the Preferred Pareto Optimal Subset Problem (PPOSP).

The PPOSP is formulated over the percentile set. There are several advantages in optimizing over the percentile set rather than the objective function space. In many real world multi-objective problems, the objective functions typically have different evaluation metrics and units. For example, objective functions can measure costs, distances or volume. There may also be a large range of values associated with the different objective functions. Normalizing and adjusting these values require application-dependent knowledge and expertise. The percentile set on the percentile space uses a ranking (ordinal) approach, which normalizes the different objective functions, comparing the relative order instead of the value of each objective function. Another advantage of working in the percentile space comes from a usability perspective. It is often much easier for a decision-maker to visualize solutions in terms of ranks as opposed to actual values. The ability to use actual values also require detailed expert knowledge of the problem, where as ordinal ranking allows for generalization. This ability to encapsulate the data for simpler representations can be beneficial to the subsequent decision process. By transforming the data into percentiles, detailed information regarding the objective function values of the solutions will be lost. See Kao and Jacobson (2008) for specific examples to illustrate these points. Kao and Jacobson (2008) also show how the PPOSP can be generalized by using different range normalization approaches.

There are two preferential parameters in PPOSP, the size of the desire subset, $N0$, and the structure of the percentile function, typically in the form of a value function (e.g., a convex combination of the objective functions). The optimal threshold percentile vector(s) for the PPOSP define(s) the preferred reduced subset of solutions, N_{sub}. Each of the threshold percentiles is analogous to the weight preferences used in the value function approach (Korhonen and Halme 1990). However, instead of manually assigning weight preferences for each objective function, this manual procedure is captured within the PPOSP, which provides a method for filtering undesirable solutions (i.e., solutions that do not satisfy the threshold values found by the PPOSP). Finding such a reduced subset of Pareto optimal solutions reduces the burden on the decision-maker to closely examine a large number of Pareto optimal solutions.

Kao and Jacobson (2008) prove that the corresponding decision PPOSP problem and the more general decision problem, without the Pareto property, are both *NP*-complete. For clarity, define ei to be a vector of size k, where all components are 0 except for the i^{th} component. A formulation of the corresponding decision problem for the PPOSP and its more general form are given.

Dominating Pareto Subset Problem (DPSP)
INSTANCE: Finite Pareto set $U \subset Z^k$, $|U| = N$, positive integer B and N', where $N' < N$.
QUESTION: Does there exist a subset $U' \subset U$, such that $\Sigma_{i=1,2,\ldots,k} \min_{u \in U'} (ei \bullet u) \geq B$ and $|U'| \geq N'$?

A more general formulation of the DPSP is to remove the Pareto restriction on the set U.

Dominating Subset Problem (DSP)
INSTANCE: Finite set $U \subset Z^k$, $|U| = N$, positive integer B and N', where $N' < N$.
QUESTION: Does there exist a subset $U' \subset U$, such that $\Sigma_{i=1,2,\ldots,k} \min_{u \in U'} (ei \bullet u) \geq B$ and $|U'| \geq N'$?

Kao and Jacobson (2008) prove that *DPSP* and DSP are NP-complete using a polynomial transformation from Max *N-M* Biclique Problem, which also had to be proven to be *NP*-complete using a polynomial transformation from the *NP*-complete Maximum Edge Biclique Problem (Peeters 2003). Kao and Jacobson (2008) show that DPSP is polynomial for $k = 2$ (i.e., for a bi-objective problem, the optimal subset N_{sub} can be found in $O(|S^{PO}| \log |S^{PO}|)$ time). To see this, sorting the solution percentile vector along a single objective function provides an ordering, which also implicitly provides an ordering for the second objective function (due to the Pareto property). Enumerating all consecutive N' subsets of the ordered set finds the optimal subset of Pareto optimal solutions (see Deterministic Sorted Local Search in Kao and Jacobson 2008).

Kao and Jacobson introduce two exact algorithms and five heuristics for finding optimal/near-optimal solutions for the PPOSP. Two different enumeration approaches are presented for solving the PPOSP. Since the threshold percentile vectors define unique subsets of Pareto optimal solutions, the PPOSP can also be solved by enumerating over all threshold percentile vectors. This enumeration, termed the Diagonal Enumeration (DE) algorithm, takes $O(|S^{PO}|^k)$ time. Alternatively, another approach is to enumerate all possible subsets of Pareto optimal solutions of size N'. This enumeration, termed the Branch and Cut (BC) algorithm, takes $O(|SPO|^{N'})$ time. Depending on the parameters S^{PO}, N', and k, the two different brute force enumerations result in different running time performances.

The DE algorithm *avoids* enumerating over all combinations of threshold percentile values. Depending on the threshold percentile vector, the corresponding subset N_{sub} may have size less than N'. In order for the percentile value function to be maximized with respect to N', the size of N_{sub} must equal N'. If $|N_{sub}| > N'$, then by reducing the size of N_{sub}, q will either remain the same or increase. Lemma 3.1 states this formally.

Lemma 3.1 (Kao and Jacobson 2008): *If $U \subset S^{PO}$ and $U' \subset U$, where* p (p') *is the corresponding threshold percentile vector associated with U (U'), then q(p) $\leq$ q(p').*

The DE algorithm exploits the results in Lemma 3.1 to avoid performing a full enumeration by constructing a k-dimensional table (called the *DE table*), where each entry within the table corresponds to a subset of Pareto optimal solutions. By design, each of the k dimensions corresponds to the k objective functions, where the indices along each of the dimensions correspond to percentile values. These indices also represent the sorted order of the percentile values (i.e., index i along dimension j corresponds to the i^{th} smallest percentile value of the j^{th} objective function.) The index of each entry can therefore be mapped to a valid threshold percentile vector.

The enumeration is done by systematically constructing the *DE table* in a diagonal manner (as illustrated in Figure 2 in Kao and Jacobson 2008). The advantage in constructing the *DE table* in such a manner is to avoid a full enumeration. If all entries along a single diagonal pass of the *DE table* fail to contain at least N' elements, then the enumeration process can be terminated, since all diagonal passes thereafter will only contain percentile vectors with larger components. Furthermore, it is unnecessary to enumerate indices along a particular dimension if the size of the corresponding subsets is less than N'. In the worst case, this algorithm will construct the entire *DE table*, and hence, the running time is $O(|S^{PO}|^k)$.

The DE algorithm solves the PPOSP by enumerating all combinations of percentile values of the threshold percentile vector. Alternatively the PPOSP can be solved by enumerating all subsets of Pareto optimal solutions of size N'. This enumeration approach is used to construct the BC algorithm. This enumeration approach can be done by constructing $|S^{PO}|$ search trees, where each node of a search tree corresponds to a subset of Pareto optimal solutions, and the root of each search tree is a unique element of S^{PO}. The second level of each of the search trees consists of all 2-element subsets constructed by adding a new element to the root. The third level consists of all 3-element subsets by adding a new element to its parent. Each level of the search trees is constructed by adding a new element to the parent. Therefore, each search tree will have at most N' levels, where if all of such search trees are fully constructed, then this corresponds to enumerating all N' subsets. The BC algorithm constructs each of the N' search trees, starting at the root. However, it avoids performing a full enumeration by deciding whether to branch or cut at each node of the different search trees. Since each node in the search tree corresponds to a subset of Pareto optimal solutions, the corresponding percentile function value can also be calculated. If at any node, the percentile function value is less than the current best percentile function value of a subset with N' elements, a cut is performed at that node and further enumeration along that branch is unnecessary, since from Lemma 3.1, any further branching along such nodes will only decrease the percentile function value.

A random subset of Pareto optimal solutions of size N' is generated for the initial best-to-date percentile function value. The higher the initial percentile function value, the less branching that is needed for the enumeration. However, in the worst case, the BC algorithm corresponds to enumerating all subsets of Pareto optimal solutions of size $N0$, and hence the worst case running time is $O(|S^{PO}|^{N'})$.

Kao and Jacobson (2008) also introduce two constructive heuristics for finding good solutions to the PPOSP. The Greedy Constructive Elimination (GC-) heuristic creates a preferred subset of Pareto optimal solutions by eliminating elements from S^{PO} until the size of the preferred subset is N'. The GC- heuristic starts by considering the full set of Pareto optimal solutions S^{PO}. It then finds a subset of S^{PO} of size N' by iteratively eliminating elements from S^{PO}. The percentile vector, which provides the best improvement over the percentile function value if it is removed, is eliminated at each iterative step. In the case of ties, a randomly selected percentile vector among the ties is eliminated. This heuristic has running time of $O(|SPO|)$. In contrast, the Greedy Constructive Expansion (GC+) heuristic builds a preferred subset of Pareto optimal solutions by adding elements to an empty set until the size of the subset is N'. The GC+ heuristic is motivated by the BC algorithm. Like the BC algorithm, it starts by considering $|S^{PO}|$ subsets of Pareto optimal solutions, each with a single distinct element of S^{PO}. However, unlike the BC algorithm, at each level in constructing a search tree, the GC+ heuristic greedily selects the best node to branch (i.e., an element is added to the current subset(parent) only if it decreases the percentile function value of the current subset the least). In the case of ties, a random solution is selected. A cut is performed, as in the BC algorithm, based on the best-to-date percentile function value. The GC+ heuristic builds $|S^{PO}|$ such search trees with distinct roots, where each search tree is a simple path of length at most N'. Since each of the elements in S^{PO} are used as the initial subsets, there could be $|S^{PO}|$ different subsets of Pareto optimal solutions of size N' (i.e., each of the $|S^{PO}|$ different search trees) . The intuition behind this heuristic is to find an optimal constructive ordering (i.e., an optimal ordering of increasing the initial subset such that the resulting subset of Pareto optimal solutions is optimal), where constructing each subset of Pareto optimal solutions takes $O(|S^{PO}|N')$ time. Since there are $|S^{PO}|$ such starting subsets, the worst case running time for the GC+ heuristic is $O(|S^{PO}|^2 N')$. Both of these heuristics use a greedy selection rule. For additional details on these heuristics, see Kao and Jacobson (2008).

Kao and Jacobson (2008) also introduce three local search heuristics. Local search heuristics are typically characterized by the following three steps:

1. Generate a feasible solution, s.
2. Attempt to find an improved feasible solution s' in a neighborhood of s.
3. If improved solution is found, replace s with s'. Repeat from Step 2.

The Deterministic Sorted Local Search heuristic examines subsets based on the sorted ordering of each objective function. This heuristic is different from the typical local search heuristic in that it uses a fixed deterministic neighborhood. The Element Exchange Local Search heuristic and the Percentile Neighborhood Local Search heuristic differ primarily in their neighborhood functions. While the Element Exchange Local Search heuristic defines its neighborhood function by altering the subset of Pareto optimal solutions, the Percentile Neighborhood Local Search heuristic defines its neighborhood function by perturbing the threshold percentile vector.

The Deterministic Sorted Local Search (DSLS) heuristic examines subsets of Pareto optimal solutions of size N' by considering percentile vectors sorted by one of the objective functions. Therefore, S^{PO} is sorted k times by each objective function (i.e., there are k different sorted ordering of S^{PO}), where each of the k sorted orderings is examined by considering subsets of size N' with consecutive elements in the sorted $S^{PO.}$ The best percentile function value found is then returned. Since traversing each sorted S^{PO} takes linear time, the sorting of S^{PO} dominates this heuristics' running time. In particular, the DSLS heuristic has running time $O(k|S^{PO}|\log|S^{PO}|)$. Lemma 3.2 shows that in a bi-objective problem, a subset of Pareto optimal solutions cannot have the maximum percentile function value unless the subset contain only elements that are consecutive in a sorted ordering based on one of the objective functions. Using this result, the DSLS heuristic finds the optimal subset of Pareto optimal solutions for the bi-objective problem.

Lemma 3.2 (Kao and Jacobson 2008): *Let* $U \subset S^{PO} \subset \Re^2$, $(u'_1,u'_2) \notin U$. *If there exists some* (u_1,u_2), $(v_1,v_2) \in U$ *such that* $u_1 > u'_1$ *and* $v_2 > u'_2$, *then the corresponding percentile function value of U cannot be the optimal.*

By the Pareto property, sorting S^{PO} based on one of the objective functions implicitly sorts the other objective function values. This ordering is a necessary condition for optimality, as shown in Lemma 3.2 for $k = 2$. Moreover, since Lemma 3.1 states that the optimal subset must be of size N', then the DSLS heuristic must find the optimal solution for the bi-objective problem.

The Element Exchange Local Search (EELS) heuristic uses a single element exchange neighborhood function. The single element exchange neighborhood function transforms a feasible subset of Pareto optimal solutions by substituting percentile vectors in and out of the current feasible subset of Pareto optimal solutions. By design, this single element exchange neighborhood function can enumerate all possible subsets of size N'. This neighborhood function is quite general and provides limited direction for the local search. To provide more restrictions and to increase efficiency of the local search, two greedy modifications are added. The first modification forces the neighborhood function to greedily select the best element for the single element exchange, which provides the largest improvement to the percentile function value of the current feasible subset of Pareto optimal solutions. The second modification limits the candidate percentile vectors considered for the feasible subsets of Pareto optimal solutions. An element that has been removed from the current subsets of Pareto optimal solutions is eliminated from any further consideration. The single element exchange neighborhood function is modified to only consider elements in the *pool*, defined as the set of candidate elements that have not been considered in any feasible subsets. These two greedy modifications significantly increase the efficiency of the EELS heuristic. Since each percentile vector can be exchanged into a feasible subset at most once, and at each iteration there are at most $|S^{PO}|$ comparisons, then the worst case running time for a single starting initial feasible subset is $O(|S^{PO}|^2)$. The single element exchange neighborhood function is of size $|S^{PO}|$. One variation of this neighborhood function is to perform multiple element exchanges. However, increasing the number of exchanges also increases the size of the neighborhood. Since the size of the neighborhood increases exponentially, greedily selecting the best percentile vector would be infeasible, although such an expanded neighborhood would reduce the number of local optima. To avoid being attracted to the same local optimum, the EELS heuristic is restarted with new random initial subsets. If the number of restarts is given by C, then the worst case running time for the EELS heuristic is $O(C\,|S^{PO}|^2)$.

The Percentile Neighborhood Local Search (PNLS) heuristic is motivated by the DE algorithm. Recall that each entry in the *DE table* corresponds to a subset of Pareto optimal solutions. The DE algorithm may enumerates many subsets of Pareto optimal solutions, with sizes much larger than N'. Lemma 3.1 shows that these subsets of Pareto optimal solutions are not optimal. The PNLS heuristic modifies the DE algorithm by avoiding enumeration of entries with corresponding subsets of size greater than N'. The neighborhood function for the PNLS heuristic maps each entry in the *DE table* to a set of neighboring entries, where an entry is then visited based on the size constraint and the percentile function value. The intuition behind this neighborhood function is that neighboring entries should correspond to subsets of similar sizes. By setting the initial entry with a corresponding subset of size N', this allows the heuristic to examine entries with corresponding subsets of similar sizes. In the worst case, this neighborhood function may enumerate the full *DE table*. The PNLS heuristic biases the neighbor selection to avoid enumerating the full *DE table*. A new neighboring entry is selected based on the size of the corresponding subset as well as the corresponding percentile function value. Subsets of size N' with improving percentile function value are considered first. The heuristic terminates when a threshold, given by T, of non-improvement neighboring searches are made. The PNLS heuristic is initialized at a starting entry where the size constraint is at equality. An entry with subset of size N' can be found by increasing the percentile value, which then can be used as the initial entry for the PNLS heuristic. This can be repeated for each of the k objective functions. Since the PNLS heuristic searches for the optimal solution in a state space of size $|S^{PO}|^k$, then it has a worst case running time of $O(|S^{PO}|^k)$, similar to the DE algorithm.

Kao and Jacobson (2008) report extensive computational results with the algorithms and heuristics described above. These computational results suggest that the GC+ heuristic and
the PNLS heuristic are the most effective in finding the optimal subset of Pareto optimal solutions. In particular, the GC+ heuristic found the optimal solutions in 60% of the experimental runs executed, while the PNLS heuristic found the optimal solutions in 75% of the experimental runs executed. The worst-case GC+/DE and PNLS/DE ratios (i.e., measures of performance for the heuristics in comparison to the optimal solutions found by the DE algorithm) were 0.995 and 0.973, respectively. Although the EENS heuristic did not find the optimal solutions, it was still very efficient, always obtaining solutions within 10% of the optimal solutions. The GC- heuristic and the DSLS heuristic always found solutions within 20% of the optimal solutions. For a complete discussion and details on these computational results, see Kao and Jacobson (2008).

Multi-objective optimization problems occur in numerous real-world applications. Solving such problems can yield a large number of Pareto optimal solutions. Kao and Jacobson (2008) examines the question of identifying preferred subsets of Pareto optimal solutions. The formulation of the discrete optimization problem, PPOSP, is designed to assist a decision-maker in finding preferred subsets of Pareto optimal solutions. The PPOSP introduces a new approach to address the post-optimality selection problem, by providing a framework that minimizes the need for expert knowledge in the decision-making process, and hence, reducing the burden on the decision-maker so as to focus their attention on preferred reduced subsets of Pareto optimal solutions. The PPOSP allows the decision-maker to obtain a desirable subset size N', based on threshold values for each objective function. Moreover, it does not require expert knowledge in finding such reduced preferred subset, which then allows the decision-maker to focus on smaller sets of preferred Pareto optimal solutions. In addition, unlike typical value function approaches, the PPOSP is formulated in (but not limited to) the percentile space, which provides in ordinal approach in addressing the post-optimality selection problem. The decision formulation of the PPOSP is formulated and proven to be *NP*-complete. Two exact algorithms, the DE algorithm and the BC algorithm, are provided for solving the PPOSP to optimality. Five heuristics are also presented, which provide a spectrum of heuristics with varying trade-offs in solution quality and run time efficiency. The experimental results reported suggest that the GC+ heuristic can yield the best results, if running time can be sacrificed. Otherwise the EELS heuristic provides the best trade-off, efficiently returning quality solutions. Note that the heuristics presented in this paper do not require the set of solutions to be Pareto. Although the decision problem for a non-Pareto set is also proven to be *NP*-complete, it is not clear what the impact of the Pareto property has on these heuristics. The Pareto property provides structure to the feasible solution set for the PPOSP. For bi-objective problems, the DSLS heuristic uses this structure to find the optimal solution. However, it is not apparent, at this time, how one can exploit such structure in higher dimensional problems. Providing higher levels of encapsulation, while retaining the consistency of the decision-maker preferences, is an area of current research activity. Another area of research is to address the scalability of the heuristics and algorithms higher dimensional problems. The ultimate goal of this effort is to design a fully automated post-optimality selection process; work is in process to design such a procedure.

4. A Sequential Stochastic Security System Design Problem for Aviation Security

Aviation security is an issue of national concern. The events of September 11, 2001, prompted multiple operational changes at all commercial airports, as well as sweeping changes in aviation security policy (Mead 2002, 2003). An important class of problems that arise in aviation security is the screening of passengers prior to boarding an aircraft. Developing strategies to effectively and efficiently screen passengers, as well as allocate and operate screening devices, can be quite challenging. Moreover, even after such systems are in place, it can be very difficult to measure their effectiveness.

The Aviation and Transportation Security Act (ATSA), enacted on November 19, 2001, by the United States Congress, transferred aviation security responsibilities from the Federal Aviation Administration (FAA) to the (newly-created) Transportation Security Administration (TSA), housed within the United States Department of Homeland Security. An important aviation security policy change that was part of the ATSA was the requirement for 100% checked baggage screening by December 31, 2002. Prior to this, only a small fraction of checked baggage was screened, based on the Commission on Aviation Safety and Security, established on July 25, 1996 and headed by (then) Vice-President Albert Gore, which recommended that the aviation industry improve security using existing explosive detection technologies, automated passenger prescreening, and positive passenger-baggage matching. Up until that time, the FAA had been working with the airlines to annually purchase and deploy explosive detection systems (EDSs) at airports throughout the United States. From 1998 until September 11, 2001, EDSs were only used to screen checked baggage of *selectee* passengers, those who were not cleared by a computer risk assessment system (i.e., the Computer-Aided Passenger Prescreening System --- CAPPS) developed in conjunction with the FAA, Northwest Airlines, and the United States Department of Justice. The checked baggage of *nonselectee* passengers (i.e., those who were cleared by such a system) received no additional security attention. There were no further security screening differences between selectee and nonselectee passengers. The 100% checked baggage screening policy eliminated the distinction between selectee and non-selectee passengers.

The primary objective of 100% checked baggage screening is to improve security operations at the nation's commercial airports. To meet this objective, the TSA is committed to develop new security system paradigms that can optimally use and simultaneously coordinate several security technologies and procedures. In 2005, there were over 650 million passengers traveling in the United States, with forecasts of nearly one billion passengers by 2015 (FAA 2006). Recent research suggests that greater scrutiny of passengers perceived as high risk (from a security standpoint) is more cost-effective. Butler and Poole (2002) suggest that the TSA's policy of 100% checked baggage screening is not cost-effective and that enhancing the binary screening paradigm to a multilevel screening system would be more cost-effective. Poole and Passantino (2003) endorse risk-based aviation security procedures, assigning passengers and baggage to security devices in proportion to their perceived risk. They suggest that multiple levels of security may be more effective than treating all passengers as indistinguishable (from a security standpoint).

The TSA further developed computerized risk assessment systems with the introduction of CAPPS II, an enhanced computer-based system for systematically prescreening passengers, which partitions passengers into three risk classes (as opposed to two classes by CAPPS), plus pays special attention to individuals on terrorist watch-lists available from government intelligence agencies. A frequently mentioned criticism of any system designed to classify passengers into risk classes, including CAPPS and CAPPS II, is that such systems can be gamed through extensive trial and error sampling by a variety of passengers through the system (Barnett 2001, Chakrabarti and Strauss 2002). Martonosi and Barnett (2006) and Martonosi (2005) note that trial and error sampling may not increase the probability of a successful attack and that CAPPS II may not substantially improve aviation security if the screening procedures for each type of passenger are not effective. Barnett (2004) suggests that CAPPS II may only improve aviation security under a particular set of circumstances and recommends that CAPPS II be transitioned from a security centerpiece to one of many components in future aviation security strategies. On July 14, 2004, the TSA announced the dismantling of CAPPS II (due to privacy concerns), despite having invested $100M into its development (Hall and DeLollis 2004). Shortly thereafter, the TSA announced the development of *Secure Flight*, which would focus exclusively on terrorist watch-lists (Singer 2004).

Several articles formulate aviation security problems as integer programming and discrete optimization models. Jacobson et al. (2001) provide a framework for measuring the effectiveness of a baggage screening security device deployment at a particular station (e.g., an airport terminal). Jacobson et al. (2003) introduce three performance measures for baggage screening security systems and use these models to assess their security impact on system design for single or multiple stations. Jacobson et al. (2005a) formulate problems that model multiple sets of flights originating from multiple stations subject to a finite amount of resources. These problems consider three performance measures, and examples suggest that one of the performance measures may provide more robust screening device allocations. Virta et al. (2002) consider the impact of originating and transferring selectee passengers on the effectiveness of baggage screening security systems. In particular, they consider classifying selectee passengers into two types; those at their point of origin and those transferring. This analysis is noteworthy since at least two of the hijackers on September 11, 2001 were transferring passengers. Babu et al. (2006) investigate the advantages of partitioning passengers into several groups, where a different screening strategy is used for passengers in each of the groups and the probability that each passenger is a threat is assumed to be constant. McLay et al. (2005) analyze checked baggage screening systems that use a prescreening system and different baggage screening devices, one to screen baggage of selectee passengers and the other to screen baggage of nonselectee passengers. McLay (2006) identifies models for designing security systems that partition passengers into several groups using discrete optimization, dynamic programming and heuristics. A problem that models sequential, stochastic passenger arrivals is considered, and an optimal screening policy is determined.

Research attention has also focused on the experimental and statistical analysis of risk and security procedures on aircraft. Barnett et al. (2001) report the results of a large-scale two-week experiment at several commercial airports to assess the costs and disruptions that would arise from using positive passenger baggage matching (an aviation security procedure) for all flights. Barmett et al. (1979) and Barnett and Higgins (1989) study mortality rates on passenger aircraft and perform a statistical analysis on these data.

Aviation security devices deployed at airport security checkpoints are used to detect prohibited items (e.g., guns, knives, explosives). Each security device provides a different level of security for passengers, and determining the types of security devices to deploy can be challenging. Moreover, once such security devices are deployed, the practical issue of determining how to optimally use them can be difficult. For an airport security system design problem, one objective is to maximize the security of passengers given an available budget. However, there are many ways to define security. For passenger screening, security may be measured by the number of prohibited items detected as a result of screening, or the probability that a given percentage of such items is prevented from being carried onboard an aircraft. In the worst case, each passenger is a threat, since they have the potential to carry a prohibited item onto an aircraft. The reasoning behind such a classification is that even if a passenger does not intend to be involved in a planned attack targeted at an aircraft, a prohibited item that is carried onto an aircraft by one passenger may be used by any other passenger on the same airplane.

A systematic approach is presented for designing a passenger and carry-on baggage screening system using stochastic optimization and sequential assignment theory (Derman et al. 1972). The two key components of passenger screening are the device allocation problem (i.e., the purchase and installation of security devices) and the passenger assignment problem (i.e., the operation of security devices), which to date have been addressed as separate problems. These problems are addressed simultaneously using models for optimally designing and operating security device systems. These models can be used to provide insights into the operation and performance of passenger screening, under the assumption that a passenger prescreening system (such as CAPPS) has been implemented and is highly effective in identifying passenger risk ((TSA 2005).

Designing an effective passenger screening system requires two stages: purchasing and installing security devices and screening passengers with these security devices. A passenger screening system's effectiveness depends on decisions made in both of these stages and reflects the ability of a security system to prevent threat items from being carried onboard an aircraft. Although purchasing and installing security devices is often done several months or years prior to their use, aviation security systems are designed based on their expected future value, estimated passenger throughput, and the average number of prohibited items that passengers may attempt to carry onto an aircraft. Similarly, screening passengers in real-time depends on the security devices that have been installed and are available.

Several definitions and terms are needed to describe the two-stage model framework. A *threat item* is any object carried by a passenger that is prohibited by the TSA. Threat items include weapons, explosives, incendiaries, and other items that may appear harmless but can be used to inflict damage on an aircraft (TSA 2004). Note, that the definition of a threat item can change based on intercepted attacks, intelligence, and the DHS color-coded threat level (for example, in August, 2006, in London terrorists intended to use liquid explosives as a means of attacking and destroying several US-bound airplanes; as a result, water bottles were subsequently classified as threat items). A *security device* is an aviation security technology and/or procedure used to identify a threat item. Examples of security devices include x-ray machines (designed to detect knives and guns in carry-on baggage), explosive trace detectors (designed to detect trace particles of explosives in carry-on and checked baggage), explosive detection systems (designed to detect explosives in checked baggage), and detailed hand search by an airport security official (designed to detect items not found by metal detectors and to resolve alarms issued by such detectors). A *set of security devices* is a group of security technologies and/or procedures that can be collectively considered for use at an airport security checkpoint. By design, a set of security devices may contain several identical security devices. A *security class* is defined by a preassigned subset of a set of security devices through which passengers are processed prior to boarding an aircraft. The *security level* of each security class is a measure based on the security procedures with each security device used to screen passengers in that security class. A prescreening system, such as CAPPS, assigns each passenger an *assessed threat value*, which quantifies the risk associated with the passenger. Passenger assessed threat values are random variables, where the assessed threat value for a specific passenger is referred to as a *realized assessed threat value*. The *annual cost* associated with a security device includes purchase, installation, maintenance and operational costs as well as the annual salary of security personnel required to operate the security device. The annual cost associated with a security device is estimated based on its expected estimated lifetime.

The security system design problem is formulated as a two-stage passenger screening model. The first stage is modeled as a deterministic problem that determines the set of security devices to purchase and install, subject to device-related feasibility constraints (e.g., budget and space constraints). The second stage is modeled as a stochastic problem that determines how to screen passengers arriving (in real-time) with the available security devices subject to passenger assignment constraints.

Consider a general representation of the security system design problem, formulated for a particular airport. In the first stage, define V as the space of all sets of security devices that satisfy the device-related feasibility constraints, and hence, each $v \in V$ is a set of security devices that can be purchased and installed in an airport. In the second stage, passengers are assigned to and screened by a set of security devices purchased and installed in the first stage.

Suppose that N passengers are expected to check-in during a given fixed time period. For hub airports in the United States, airport security resources are allocated based on peak-period passenger throughput (i.e., the average passenger volume during certain hours (usually 4-12 hours) of regular airport security operation on any typical day). The idea behind this assumption is that if a certain level of security can be provided during a peak-hour of airport operation, then all other operational period passenger volumes can be handled as well. The few exceptions (such as periods around major holidays) are handled on a case-by-case basis, by allowing more peak-hours and/or scheduling extra security personnel on such days.

Define A as the set of all feasible passenger assignments, and hence, $a \in A$ is an N-dimensional vector of assignment variables. A passenger prescreening system is available to provide passenger risk assessments, and hence, it can be used to determine the assessed threat value for each passenger (i.e., the realized assessed threat value). The passenger assessed threat values are given by the random vector $AT \in [0,1]^N$. For a given set of security devices $v \in V$, a realization of the assessed threat vector $at \in [0,1]^N$, and a passenger assignment vector $a \in A$, the function $G(a, v, at)$ measures the level of security. The objective is to identify a set of security devices $v^{opt} \in V$ that maximizes $E_{AT}[\max_{a \in A} G(a, v, AT)]$. Therefore, the two-stage model is given by

$$\max_{v \in V} E_{AT}(\max_{a \in A} G(a,v,AT)), \tag{4.1}$$

which is an optimization problem embedded within an optimization problem. The objective of the first stage is to identify a set of security devices that maximize an objective function, which is a function of the solution of the second stage (i.e., passenger assignments). The objective of the second stage is to determine a passenger assignment for each realization of the assessed threat vector that maximizes an objective function, which is a function of the solution of the first stage (i.e., the set of security devices). Therefore, the two stages are interdependent, and hence, must be addressed accordingly.

Several papers model the purchase and installation problem associated with security devices. Jacobson et al. (2005b) present NP-complete decision problems that capture the deployment and utilization of baggage screening security devices. McLay et al. (2006) and McLay and Jacobson (2007a,b) formulate problems that consider budget allocation based on security device costs, and introduce knapsack problem variations to address them. A greedy heuristic is introduced that obtains approximate solutions. Other research focuses on assigning passengers to two or more security classes. McLay (2006) considers the real-time operation of passenger screening systems by formulating a passenger assignment problem as a Markov decision process and shows how the optimal policy can be obtained by dynamic programming. McLay et al. (2007) solve a deterministic passenger assignment problem using integer and linear programming models.

To summarize, Jacobson et al. (2005b), McLay et al. (2006), and McLay and Jacobson (2007a,b) assume deterministic distributions of passenger risk levels for device allocation, which corresponds to the first stage of (4.1). Moreover, McLay (2006) and McLay et al. (2007) assume a given set of available security classes for passenger assignment, which corresponds to the second stage of (4.1). This research combines these two problems such that an optimal decision/policy is made at each of these stages, making the resulting problem much more realistic but also more challenging to solve.

The first stage of the model considered (device allocation) answers two questions: what set of security devices should be installed, and what security classes should be selected given this set of security devices? Moreover, different security classes are allowed to share security devices, which is not the case in Jacobson et al. (2005b), McLay et al. (2006,2007), and McLay and Jacobson (2007a,b). Lastly, with the induced stochasticity in passenger assessed threat values, the formulated model is a close approximation to the way in which airport security checkpoints are designed and used. The model also takes into account an implicit dependence between purchasing and installing security devices and assigning passengers to such security devices. A solution to (1) answers the question of how many security devices of each available type should be purchased and installed, as well as providing a passenger assignment strategy to optimally utilize these security devices.

The Sequential Stochastic Security Design Problem (SSSDP) is introduced, which models the screening operations of passengers and carry-on baggage in an aviation security system. A solution to this problem provides an optimal set of security devices to purchase and install, and for a given set of passengers, the optimal assignment of each passenger to these security devices. To describe SSSDP, consider an airport security checkpoint where security devices are to be installed and operated, and a fixed time period during which the passenger arrival rate to the security area in the terminal can be assumed to be constant (e.g., during a peak-hour of operation). Assume that the total number of passengers expected to check-in during this time period is fixed, and that the space requirements and the security device capacities are given. Upon check-in, a passenger's assessed threat value becomes known (i.e., realized), and the passenger is assigned to a security class. To formulate SSSDP, the following sets, variables, parameters and functions are needed,

N - number of passengers to enter the airport security checkpoint over a given fixed time period,
D - set of security devices available,
c_d - capacity of security device $d \in D$,
s_d - space at an airport security checkpoint required for security device $d \in D$,
M_d - annual cost associated with security device $d \in D$,
J - a set of security classes, defined as a family of unordered subsets of D,
L_j - security level associated with security class $j \in J$,
S - space available at an airport security checkpoint to install and operate security devices,
B - annual budget available to purchase, install and operate security devices,
$\boldsymbol{AT}$ - passenger assessed threat (random) vector, $\boldsymbol{AT} \in [0,1]^{N}$, where AT_i is the assessed threat value (random variable) for passenger i=1, 2,…,N,
$P_{AT}(at)$ - probability mass function of assessed threat vector $\boldsymbol{AT}$.
I_{dj} = 1 if class $j \in J$ contains device $d \in D$, 0 otherwise
X_j - number of passengers assigned to security class j 2 J (decision variable),
Y_d - number of security devices of type $d \in D$ purchased and installed (decision variable).

Security device parameters are needed to formulate device-specific feasibility constraints in the first stage of (4.1). The capacity of a security device is defined as the upper bound on the hourly rate at which passengers can be screened by the device. The space parameter of a security device is defined as the area (in square meters) required by the security device when installed and in operation. The annual cost is the amount of resources (per year) required to purchase, install and operate a security device. The budget is the total amount of resources available annually for purchasing, installing and operating the security devices and screening passengers. Security level L_j is defined as the conditional probability of detecting a threat item by security class j given that a passenger is carrying a threat item. This probability, known as the *true alarm* rate, is a function of the detection probabilities associated with the devices and the procedures used to screen passengers in security class j.

The assessed threat values quantify the risk associated with each passenger. The assessed threat value of a passenger can be defined as the probability that the passenger is carrying a threat item. The realized assessed threat values are based on information that passengers provide, either implicitly or explicitly, which is processed (in real-time) through an automated system such as Secure Flight. Assume that the assessed threat values for each passenger are independent, and that the realized assessed threat values are accurate representations of passengers' true level of risk.

Assume that each passenger is carrying either zero or one threat item. Then, for a given set of passengers, the expected number of threat items detected is the sum (over the set of passengers) of the products of each passenger's realized assessed threat value and the security level of the respective security class to which the passenger is assigned; let this sum be the objective function $G(a,v,at)$ in the second stage in (1). Then the second stage can be modeled as an instance of the sequential stochastic assignment problem [9]. By definition, for a given passenger assessed threat distribution, the expected number of threat items that enter the security area is $\Sigma_{i=1,2,\dots,N} E(AT_i)$.

For a set of security levels of available security classes $\prod_{j\in J} \{L_j\}^{Xj}$ and a realized assessed threat vector ***at***, define the *stochastic sequential assignment function* $G = f_{SSA}(\prod_{j\in J} \{L_j\}^{X}$ ***at***) as a linear function that returns the expected number of threat items detected when an optimal, sequential assignment policy is followed in the second stage. Then, the SSSDP is formulated as

$$\begin{aligned} &\max E_{AT}(f_{SSA}(\textstyle\prod_{j\in J}\{L_j\}^{Xj}, \mathrm{P_{AT}})) && (4.2)\\ &\text{subject to} \quad \Sigma_{j\in J} X_j = N \\ &\qquad \Sigma_{j\in J} I_{dj} X_j \le Y_d c_d,\ d\in D \\ &\qquad \Sigma_{d\in D} M_d Y_d \le B \\ &\qquad \Sigma_{d\in D} s_d Y_d \le S \\ &\qquad Xj \in Z^{+}, Y_d \in Z^{+}, j\in J,\ d\in D \end{aligned}$$

In (4.2), the expected number of threat items detected is maximized when assigning the N passengers to security classes. This value is given by the optimal passenger assignment policy as a function of the security classes and underlying probability mass function of the assessed threat vector. A solution to (4.2) determines the number of passengers to be screened by each security class, and assigns each passenger individually to a security class upon check-in. The first constraint ensures that all passengers are screened. The second set of constraints ensures that a sufficient number of devices of each type are purchased to assign all N passengers. The third and fourth constraints ensure that the annual costs and space required to purchase, install, and operate the security devices do not exceed the available budget and space, respectively. Note that solving the second stage in (4.1) reduces to maximizing the value expressed by $G= f_{SSA}(\prod_{j\in J} \{L_j\}^{Xj}$, ***at***). The optimal solution to the second stage of SSSDP is given in terms of a policy (see Derman et al. 1972). Theorem 4.1 formally describes how the optimal passenger assignment policy is implemented.

Theorem 4.1. (Derman et al. 1972): For each $n \ge 1$, there exist real numbers $0 \le t_{0,n} \le t_{1,n} \le t_{2,n} \le \dots \le t_{n,n} = 1$ such that whenever there are n passengers to assign to security classes with security levels $L_1 \le L_2 \le \dots \le L_n$, then the next passenger to check-in is optimally assigned to class i if the realized assessed threat value at_1 is contained in the interval $(t_{i-1,n}, t_{i,n}]$. Furthermore, $t_{i,n}$, $i=1,2,\dots,n-1$, is the expected value, whenever there are (n-1) passengers to assign, of the quantity which is assigned to the i^{th} least secure class (assuming an optimal policy is followed).

To determine the optimal assignment for the N passengers arriving sequentially at an airport security checkpoint, Theorem 4.1 must be applied (sequentially, in reverse) N times, for $n=N, N-1,\dots,1$. In particular, suppose that there are n passengers scheduled to arrive, and hence, need to be assigned to security classes, with the n security classes available indexed in order of increasing security levels. When a passenger arrives to the airport security checkpoint and checks-in, their realized assessed threat value is determined by a prescreening system. The decision to assign this passenger to one of the available security classes is made at this time (with the assessed threat values of the remaining n-1 passengers treated as i.i.d. random variables). Theorem 4.1 states that the interval (0,1] can be divided into n subintervals $(t_{i-1,n}, t_{i,n}]$, $i=1,2,\dots,n$, such that each passenger's security class assignment is determined by the subinterval that contains the passenger's realized assessed threat value (i.e., if this value is contained in the interval $(t_{i-1,n}, t_{i,n}]$, then the passenger is assigned to security class i.) Once such an assignment is made, this security class is no longer available, the number of passengers to arrive is decremented by one, and Theorem 4.1 is reapplied to determine the security class assignment for the next passenger to arrive and check-in, with the endpoints of the subintervals $(t_{i-1,n}, t_{i,n}]$, $i = 1,2,\dots,n$ recomputed (see Lemma 4.1 for the method by which these values are determined).

Given a set of security levels $\prod_{j\in J}\{L_j\}^{Xj}$, the maximum objective function value in (2) is achieved when the passenger with the smallest realized assessed threat value is assigned to the least secure class, the passenger with the second smallest realized assessed threat value is assigned to the second least secure class, and so on (for a proof of this result, see Hardy et al. 1952). Let k_i, $i=1,2,\dots,N$, be the i^{th} smallest number of threat items expected in the passenger set (see Lemma 4.1). Then, SSSDP is formulated as an integer program with binary assignment variables, defined as $A_{ij} = 1(0)$ if passenger i is (not) assigned to security class j for $j \in J$, $i = 1,2,\dots,N$. The objective is to maximize the expected number of threat items detected, which is represented by the sum of the products of the security levels and k_i, $i = 1,2,\dots,N$. With the added notations, (4.2) is transformed into

$$\begin{aligned} &\max \Sigma_{i=1,2,\dots,N} (\Sigma_{j\in J} L_j A_{ij} k_i) && (4.3)\\ &\text{subject to} \quad \Sigma_{j\in J} A_{ij} = 1,\ i=1,2,\dots,N \\ &\qquad X_j - \Sigma_{i=1,2,\dots,N} A_{ij} = 0, j\in J \\ &\qquad \Sigma_{j\in J} I_{dj} X_j \le Y_d c_d,\ d\in D \\ &\qquad \Sigma_{d\in D} M_d Y_d \le B \end{aligned}$$

$\Sigma_{d\in D} s_d Y_d \leq S$

$A_{ij} \in \{0,1\}, X_j \in Z^{+}, Y_d \in Z^{+}, j \in J, d \in D$

In (4.3), the constraints in (4.2) are reformulated under the added notation. The first set of constraints ensures that each passenger is screened exactly once. The second set of constraints ensures that the number of passengers screened by class $j \in J$ is equal to $\Sigma_{i=1,2,\ldots,N} A_{ij}, j \in J$. The remaining constraints are the same as the constraints in (4.2).

Lemma 4.1 shows how $t_{0,n}, t_{1,n}, t_{2,n}, \ldots, t_{n,n}, n = 1,2,\ldots,N$, and coefficients $k_i, i = 1,2,\ldots,N$ can be computed. In particular, it uses the probability distribution for the passenger assessed threat values to compute the intervals that determine which security class a passenger is assigned to when they check-in, based on their particular realized assessed threat value.

Lemma 4.1. (Derman et al. 1972): *Let $F_{AT}(z)$ denote the cumulative distribution function of the assessed threat value for passenger i=1,2,…,N. Define $t_{0,n}=0$, $t_{n,n}=1$ for n=0,1,…;N. Then*

$$t_{i,n+1} = \int_{t_{i-1,n}}^{t_{i,n}} z dF_{AT}(z) + t_{i-1,n} F_{AT}(t_{i-1}, n) + t_{i,n}(1 - F_{AT}(t_{i,n})) \text{ for } n=0,1,\ldots,N \text{ and } i=1,2,\ldots,n.$$

The expectation of the assessed threat value, which is to be assigned to the i^{th} least secure class, is given by $k_i = t_{i,N+1}$, i=1,2,…,N. Using these values, SSSDP is modeled as an integer program (4.3). SSSDP is now formally stated using this notation.

Stochastic Security System Design Problem (SSDDP).

INSTANCE: Positive integers N, S and B; finite set D, for each $d \in D$ $M_d \in Z^{+}$, $c_d \in Z^{+}$ and $s_d \in Z^{+}$; a finite family J of subsets of D, for each $j \in J$ $L_j \in R^{+}$; for each i=12,…,N, a positive real number k_i.

QUESTION: Is there an assignment of binary variables A_{ij}, i=1,2,…,N, $j \in J$, and a non-negative integer Y_d, $d \in D$, such that $\Sigma_{i=1,2,\ldots,N} \Sigma_{j\in J} L_j A_{ij} k_i$ is maximized and $\Sigma_{j\in J} A_{ij} = 1$, i=1,2,…,N, $\Sigma_{i=1,2,\ldots,N} \Sigma_{j\in J} I_{dj} A_{ij} \leq Y_d c_d$, $d \in D$, $\Sigma_{d\in D} M_d Y_d \leq B$, $\Sigma_{d\in D} s_d Y_d \leq S$?

Nikolaev et al. (2007) prove SSSDP to be NP-hard using a polynomial Turing reduction from the integer knapsack problem. Nikolaev et al. (2007) also provide an example to illustrate how SSDDP can be applied to actual aviation security systems, using data extracted from the Official Airline Guide (OAG) for domestic flights of a single airline carrier at the terminal of a hub airport in the United States. The data provided by the OAG include the set of flights, the number of available seats on each flight, and the departure time of each flight. Note that the aviation security data available in the public domain are of limited quality. Passenger classification is considered security-sensitive information. Moreover, performance characteristics of the latest security devices are classified information. Lastly, political forces on the aviation security community make them reactive to ongoing threat events. Given all these constraints, an effort is needed to find solutions to new threats and problems before they surface. These limitations do not reduce the value of aviation security research, but, rather, enhance it, since optimization models can be tested using new, better quality data as they become available. In essence, as data quality improves, model sophistication will become more critical and of greater interest to the TSA and other aviation security stakeholders.

In conclusion, passenger and carry-on baggage screening is a critical component of any aviation security system operation, and an important national homeland security concern. There are several possible future research directions. First, taking passenger convenience issues into account would add a new dimension to the model and analysis. In particular, SSSDP does not take into account the expected time passengers wait in security lines. Introducing waiting times suggests another class of security design problems, which could incorporate elements of queueing theory. Second, minimizing the overall false alarm rate can be useful since the majority of passengers are not threats and high false alarm rates are expensive to the airlines. However, different passenger assignment policies do not significantly impact false alarms (since false alarms need to be resolved just as frequently for low-risk passengers as for high-risk passengers), and, hence, the false alarm problem can itself be formulated only for security devices, without regard for passenger assignment. Moreover, more accurate security device data would be required to effectively use such models. Third, an extension to SSSDP can be considered where each passenger's assessed threat value dynamically changes as a result of each subsequent screening by a security device. Such multi-level dynamic assignment problems are of particular interest to the TSA (Kahn and Robinson 2006). Work is in progress to design algorithms and heuristics for all these model extensions.

5. Sequential Stochastic Passenger Screening Problems

Integer programming and discrete optimization models have been used to formulate several aviation security problems when a passenger prescreening system is used. As previously discussed, Jacobson et al. (2001) provide a framework for measuring the effectiveness of baggage screening security device deployments for screening selectee baggage at a particular airport station. Jacobson et al. (2003) introduce three performance measures for baggage screening security systems and introduce models to assess the security effect for single or multiple airport stations. Jacobson et al. (2005a) formulate problems that model multiple sets of flights originating from multiple airport stations subject to a finite amount of security resources; these problems consider the three performance measures introduced in Jacobson et al. (2003). Examples are presented to illustrate strategies that may provide more robust device allocations across all these performance measures. Jacobson et al. (2005b) construct integer programming models for problems that consider multiple sets of flights originating from multiple airports. Virta et al. (2002) consider the impact of originating and transferring passengers on the effectiveness of baggage screening security systems. Both these papers consider classifying selectees into two types; those at their point of origin and those connecting through a hub airport. This is noteworthy since at least two of the hijackers on September 11, 2001 were connecting passengers. Babu et al. (2006) use linear programming models to investigate the benefit acquired from using multiple risk groups for screening passengers. They conclude that using multiple risk groups is beneficial for security, even when a prescreening system is not used to differentiate passenger risk. Nie et al. (2006) extend this model to consider passenger risk levels, as determined by a passenger prescreening system, and formulate the resulting model as a mixed integer program. They find that using passenger risk levels results in a more efficient security system.

Optimal risk-based passenger screening must operate in real-time and be dynamic, responding to changes in passenger arrival rates and device utilization. McLay et al. (2009) introduces the Sequential Stochastic Passenger Screening Problem (SSPSP) that models passenger screening strategies using Markov decision processes and discrete optimization models. SSPSP assumes that passengers are classified as selectees or nonselectees based on the output of a prescreening system. SSPSP is motivated by McLay et al. (2006, 2007), who introduce the Multilevel Allocation Problem (MAP) and the Multilevel Passenger Screening Problem (MPSP). In these problems, multiple security classes are available for screening passengers, which generalizes the binary paradigm of Secure Flight. Moreover, the set of passengers to be screened at a particular station in an airport in a given period of time is assumed to be known, and hence, the assessed threat values are assumed to be known a priori. This assumption is relaxed by SSPSP, in which passengers check in sequentially, and each passenger's assessed threat value becomes known only upon check-in. This necessitates a change in the solution methodology since all passenger screening decisions are made simultaneously in MAP and MPSP, whereas passenger screening decisions are made sequentially for SSPSP. MPSP is a static model that does not account for passenger check-in order, whereas SSPSP incorporates the effect of passenger check-in order. Since passengers are classified as selectees or nonselectees upon check-in, it is critical to understand the impact of passenger order on the ability of a screening system to systematically identify high-risk passengers in real-time to focus more effective screening technologies on these passengers. Note that this research assumes that a prescreening system such as CAPPS has been implemented and is effective in identifying passenger risk (i.e., the assessed threat values accurately quantify passenger risk) (Kahn 2006).

The primary contribution of this research is to identify a real-time methodology for screening passengers (or in fact any objects that are attempting to enter and harm a secure area) under a binary screening paradigm and to show how this methodology can be used to provide insights into the operation and performance of such real-time systems. This research focuses on the theoretical issues surrounding this methodology in order to understand its fundamental properties, and the results provide an optimal policy for screening passengers in real-time. Providing strategies to screen passengers is critical in the design and development of next-generation aviation security systems, given the political force being exerted on the aviation security community to react to existing and new terrorist threats.

Efficiently and effectively screening passengers using a risk-based system is challenging. SSPSP models a passenger screening problem when passengers are classified as either selectees (S: high risk) or nonselectees (NS: low risk). SSPSP is formulated as a stochastic optimization model and uses a Markov decision process model to solve for its optimal policy.

Passengers and their baggage are screened by a sequence of devices, which is set in advance by the TSA for both the selectee and nonselectee classes. Note that each device gives one of two possible responses: an alarm or a clear, and hence, the system gives one of two possible outcomes: an alarm or a clear, which is a function of the device outcomes and can be defined in several ways (see Kobza and Jacobson 1996, 1997 for a discussion on how this can be done). Each passenger is either a threat or a nonthreat, where a threat is defined as a passenger carrying a prohibited item (e.g., gun, knife) through a security checkpoint. Ideally, the system yields a clear response for all nonthreat passengers and an alarm response for all threat passengers. Although it is not known in advance whether a given passenger will yield an alarm response, it is assumed that there are procedures in place for resolving alarms and that adequate resources are available to resolve all alarms given by the system.

Several assumptions are needed to define SSPSP. First, SSPSP assumes that passengers check in sequentially, and that each passenger is assigned to a class upon check-in (i.e., before the next passenger checks in). Note that the passengers may check in several hours before they arrive at the airport. Therefore, the time period when passengers are assigned to classes may be different than the time period when passengers arrive at the security area in an airport terminal. In practice, a prescreening system such as CAPPS determines which passengers are selectees using information provided by passengers at the point of ticket purchase, and hence, assigning passengers to classes when they check is consistent with existing TSA procedures and practices (US GAO 2005).

The given time interval for screening passengers can be divided into T stages, where during stage t, a passenger checks in with probability p, $t = 1,2,\ldots,T$, and hence, with probability $1-p$, no passenger checks in during stage t. If the length of each stage is sufficiently small, then it is reasonable to assume that at most one passenger checks in during each stage. Stages are defined to reject the sequential way passengers check in. The total number of stages T captures the time interval when passengers are screened, where a day is divided into several such time intervals. Let $A(t)$ denote
the assessed threat value of passenger t (a random variable), with value unknown until the passenger checks in. The probability density function of the assessed threat values, $f_A(\alpha)$, is identical for all passengers $t = 1,2,\ldots,T$. The assessed threat value of passenger t becomes known upon the passenger's arrival, taking on value $\alpha(t)$, $t = 1,2,\ldots,T$. Without loss of generality, define $\alpha(t) = 0$ if no passenger arrives during stage t. Passenger t refers to the passenger that checks in during stage t or the dummy passenger with $\alpha(t) = 0$, and hence, SSPSP can be formulated as if T passengers always check in. The capacity of the nonselectee class is defined as T; this ensures that there is sufficient capacity to classify all passengers as nonselectees.

SSPSP is formulated as a stochastic optimization problem, where the objective is to determine the optimal policy for assigning passengers to classes as they check in. A policy π defines a rule for assigning each passenger to a class, which may change after each passenger assignment is made and the state (i.e., the remaining capacity in the selectee class). A policy may be deterministic (i.e., the policy always assigns a passenger to the same class given the identical time and state) or random (i.e., the policy may assign a passenger to different classes given an identical time and state). It may also be Markovian (i.e., the policy only depends on the current passenger and current state) or history-dependent (i.e., the policy depends on the passenger assignments of the previous passengers). The optimal policy maximizes the expected security subject to assignment and capacity constraints. Security can be defined to capture several criteria, and hence, the objective is defined in a flexible way to potentially address a number of objectives.
SSPSP classifies passengers as either selectees or nonselectees. A *passenger assignment* refers to the class to which a passenger is assigned. SSPSP is formally stated as a discrete optimization problem.

The Sequential Stochastic Passenger Screening Problem (SSPSP)
Instance:

T stages,

p, the probability that a passenger checks-in during stage $t = 1,2,\ldots,T$,

$f_A(\alpha)$, a probability density function that describes the distribution of the passengers' assessed threat values,

$0 < \alpha \leq 1$,

c, the capacity of the selectee class over the entire time interval, (the capacity of the nonselectee class is assumed to be T),

LS, and LNS, the *security levels* associated with the selectee and nonselectee classes.

Variables:

$A(t)$, the assessed threat value of passenger t,

$x_S(t)$, passenger t assignment as either a selectee ($x_S(t) = 1$) or a nonselectee ($x_S(t) = 0$).

Objective: Find the policy π that determines passenger assignments $x^{\pi}_S(1)$, $x^{\pi}_S(2)$, …, $x^{\pi}_S(T)$ such that the number of passengers classified as selectees is within capacity (i.e., $\Sigma_{t=1,2,\ldots,T}\, x^{\pi}_S(t) \leq c$) and the expected total security is maximized, $z = \sup_{\pi\in\Pi} (E[\Sigma_{t=1,2,\ldots,T}\, L_S A(t) x^{\pi}_S(t) + \Sigma_{t=1,2,\ldots,T}\, L_{NS} A(t)(1-x^{\pi}_S(t))])$.

The realized assessed threat values are determined by a prescreening system such as CAPPS. The details of CAPPS are classified, and the assumptions of how prescreening systems operate are based on information disseminated in the public domain (US GAO 2005). It is also assumed that each realized assessed threat value is determined independently and in real-time, since the prescreening system is an expert system that outputs the deterministic, realized assessed threat values based on information that passengers provide at the point of ticket purchase.

The security levels can be obtained using information and data available from the TSA. The security level of each class (scaled between zero and one) is based on the security procedures of each device used to screen passengers. Security can be defined in several ways. In this research, the security levels of the selectee and nonselectee classes are measured as the conditional true alarm probability, the conditional probability that a passenger with a threat item is detected given that they are classified as selectees or nonselectees, respectively. If each assessed threat value is defined as the conditional probability that a passenger carries a threat item, then the resulting expected security maximizes the number of true alarms detected by the system.

This objective function is identical to minimizing the expected number of false clears. For simplicity, it is assumed that a passenger carries at most one threat item. Security screening devices are designed to detect one class of threat items; for example, metal detectors are designed to detect guns and knives, while EDSs and ETDs are designed to detect explosives and bombs. Each device is not equally proficient at detecting each type of threat item; however, the formulation is realistic if the parameters are defined to restrict the search to one class of threat item (e.g., explosives).

Security devices may require significant amounts of space in airport lobbies or terminals. SSPSP does not explicitly incorporate space requirements of screening devices. Secondary screening is performed for passengers who receive an alarm response. A certain number of alarms will occur, given the capacity of the selectee class, and it is assumed that the necessary security devices are available for resolving these alarms. However, since it is assumed that the capacity of the selectee class is based on the necessary number of screening devices allowed by the physical space available, then space requirements are implicitly captured by SSPSP.

There have been limited research efforts that focus on real-time, risk-based passenger screening strategies given a prescreening system such as CAPPS. SSPSP has similarities to three other types of passenger screening methodologies. First, SSPSP is a particular case of the Sequential Stochastic Multilevel Passenger Screening Problem (SSMPSP) (McLay 2006), which generalizes the binary paradigm of CAPPS to consider multiple classes for screening passengers. SSMPSP is NP-hard, and as a result, there are additional challenges in its implementation that do not apply to SSPSP. Therefore, although the more general results for SSMPSP also apply to SSPSP, the results considered in this research provide more practical, tailored recommendations for a binary screening system (i.e., selectee and nonselectee passenger classification). Moreover, the analysis of SSPSP provides insights that may be of practical value, since the TSA has repeatedly embraced the binary CAPPS screening paradigm since September 11, 2001. Lee et al. (2009) applies control theory to SSMPSP to optimally screen passengers, and hence, uses different methodologies to obtain different insights into a similar passenger screening problem.

SSPSP is similar to the second stage of a stochastic two-stage model (Nikolaev et al. 2007). The two-stage model uses discrete optimization to determine which security devices to purchase and install as well as a passenger screening strategy that maximizes device utilization. Although there is overlap between SSPSP and the two-stage model presented by Nikolaev et al. (2007), the two-stage model provides insight into issues of long-term device allocation rather than the more immediate issue of how to optimally screen passengers in real-time. Moreover, the analysis in Nikolaev et al. (2007) focuses on the aggregate level, such as analyzing expected system performance, as opposed to focusing on implications of individual passenger screening decisions (see Sections 4 and 5).

Since SSPSP has a sequential structure and makes screening decisions based only on the current state of the system, it is natural to formulate SSPSP as a Markov decision process (MDP). This MDP formulation also illustrates how the optimal policy is found. SSPSP can be formulated as an MDP with $T+1$ stages, with the state in stage $t = 1,2,\ldots,T$ describing the system before the first t passengers have been assigned to classes, where stage 1 corresponds to the initial stage, with stage $T+1$ giving the final state after all passengers have been assigned to classes.

Let S denote the set of states. Let state $s(t) \in S$ represent the remaining capacity in the selectee class before passenger $t = 1,2,\ldots,T$ has been assigned to a class. The initial state corresponds to the initial capacity, $s(1) = c$, and denotes the state in stage t as the *remaining capacity* of the selectee class, $\overline{c}$. Therefore, $|S| = c + 1$.

For passenger t, there are two actions available: classify the passenger as a selectee, if there is available capacity, or classify the passenger as a nonselectee. Given state $s(t) = \overline{c}$, if $\overline{c} > 0$, then the next passenger may be classified as either a selectee or nonselectee, and if $\overline{c} = 0$, then the next passenger must be classified as a nonselectee, $t = 1,2,\ldots,T$, for all $s(t) \in S$. The transition probabilities, which determine state $s(t+1)$ given state $s(t)$ and the passenger assignment, are defined as $p(s(t+1) \mid s(t), x_S(t)) = 1$ if $s(t+1) = s(t) - x_S(t)$, 0 otherwise, for $t = 1,2,\ldots,T$.

The objective function value of SSPSP is determined by accruing a reward after each stage in the MDP. Define the reward for classifying passenger t as a selectee or nonselectee given state $s(t) \in S$, $t = 1,2,\ldots,T$, as

$$r(s(t), A(t), x_S(t)) = L_S A(t) x_S(t) + L_{NS} A(t)(1 - x_S(t))$$

$t = 1,2,\ldots,T$. Given that passenger t has realized assessed threat value $\alpha(t)$, then the reward for classifying passenger t as a selectee (nonselectee) becomes $L_S\alpha(t)$ after passenger t checks in, and $L_{NS}\alpha(t)$ otherwise. Note that if no passenger arrives during stage t, and hence, $\alpha(t) = 0$, then the reward is zero.

The *expected total security* for SSPSP is determined by policy π, which describes the decision rule for selecting an action (i.e., the class to which each passenger is assigned) in each state and at each stage. Let $S(t)$ denote the random variable corresponding to the state before passenger t has been assigned to a class, and let

$X^{\pi}_S(S(t))$ denote the random variable corresponding to the class to which passenger t is assigned given policy π. Given that the system is initialized in state $s(1)$, then the expected total security for SSPSP is defined as

$E^{\pi}(s(1)) = E^{\pi}[\Sigma_{t=1,2,\ldots,T}\ r(S(t),A(t),X^{\pi}_S(S(t))) \mid S(1) = s(1)]$

where $r(S(t),A(t), X^{\pi}_S(S(t)))$ is the random variable corresponding to the security obtained in time period t in state $S(t)$ with decision variables $X^{\pi}_S(S(t))$ based on policy π. The objective of SSPSP is to find the optimal policy π^* such that $E^{\pi^*}(s(1)) = \sup_{\pi} E^{\pi}(s(1) \mid \pi)$.

Define the value function in stage $t = 1,2,\ldots,T$ as the optimal expected total security for assigning passenger t and the remaining T-t passengers to classes if $s(t) > 0$,

$V_t(s(t)) = \max_{x_S(t) \in \{0,1\}} \{E[r(s(t),A(t),x_S(t)) + V_{t+1}(s(t) - x_S(t))]\}$

for $t = 1,2,\ldots,T$, where for SSPSP,

$V_t(s(t)) = \max_{x_S(t) \in \{0,1\}} \{E[L_S A(t)x_S(t) + L_{NS}A(t)(1\text{-}x_S(t)) + V_{t+1}(s(t) - x_S(t))]\}$

for $t = 12,\ldots,T$, with boundary conditions (that are also known as the *optimality equations*)

$V_{T+1}(s(t)) = 0,\ s(t) \in S$

Passenger t is classified as a selectee if

$L_S\alpha(t) + V_{t+1}(s(t) - 1) \geq L_{NS}\alpha(t) + V_{t+1}(s(t))$.

McLay et al. (2009) show that the optimal policy π^*, which solves the optimality equations, can be found using dynamic programming. The total security of a SSPSP instance, given realized assessed threat values $\alpha(1), \alpha(2), \ldots, \alpha(T)$ and their assignments $x_S(1), x_S(2), \ldots, x_S(T)$, is $\Sigma_{t=1,2,\ldots,T}\ \alpha(t)(L_S x_S(t) + L_{NS}\ (1\text{-}x_S(t))$. The optimal expected total security is captured by $V_1(s(1))$.

Integration is performed when computing the value functions defined above, since the value function is an expected value. Let O(I) denote the time complexity required to perform integration at each stage in the recursion. Note that this time complexity is O(1) if there is a closed form expression for the integrand, and that, in general, I depends on the number of grid points if numerical integration is used. If the probability density function is discrete, and each assessed threat value may take on one of W values, then $I = W$. Note that since c is bounded by T, then the optimal policy can be determined in $O(T^2I)$ time and $O(T^2)$ space.

McLay et al. (2009) report several structural properties of SSPSP. In addition, MDP formulation are derived and the relationship between SSPSP and the Dynamic and Stochastic Knapsack Problem (DSKP) and the Sequential Stochastic Assignment Problem (SSA) are made explicit. First, it is noted that the optimal policy is deterministic and Markovian. DSKP and SSA are then used to illustrate the remaining structural properties for SSPSP.

Theorem 5.1: *This optimal policy for SSPSP is deterministic and Markovian.*
Proof: Follows from the number of states being finite (see Puterman 1994).

McLay et al. (2009) show that SSPSP can be formulated as a particular case of DSKP (Papastavrou et al. 1996). In the general instance of DSKP, there is a knapsack with capacity c. There is a time interval with T + 1 stages, with the probability p of an item arriving in any of the first T stages, and a probability of 1-p of no item arriving. Each arriving item has weight W and profit R, which become known upon the item's arrival. Both W and R are independently and identically distributed random variables, and they follow probability density functions that are known in advance. If an item arrives, then one of two actions must be taken: either the item is accepted (i.e., placed in the knapsack) or rejected. The objective is to maximize the expected total reward accumulated over the T + 1 stages while remaining capacity feasible. Let $EV_t^{\bar{c}}$ denote the optimal expected reward accumulated from (and including) time period t until time period T +1 with remaining capacity $\bar{c}$, $t = 1,2,\ldots,T+1$, $\bar{c} = 0,1,\ldots,c$. The optimal expected reward over all stages is given by $EV_1^{\bar{c}}$.

McLay et al. (2009) show that SSPSP is equivalent to the particular case of DSKP when all of the item types have equal weights, with w = 1 for all arriving item types. In this case, the knapsack capacity is analogous to the capacity of the selectee class and each item's reward is analogous to the assessed threat values, scaled by the difference between the security levels of the selectee and nonselectee classes, $R = A(t)(L_S - L_{NS})$. Items in the knapsack correspond to passengers classified as selectees. Rejected items correspond to passengers classified as nonselectees. Solving SSPSP by formulating it as a DSKP instance finds the optimal expected additional security acquired from classifying passengers as selectees, compared to classifying all passengers as nonselectees.

The optimal policy for DSKP is a threshold policy, in which arriving items whose weight and value are above (below) a threshold value are accepted (rejected). Define the expected reward from stage t + 1 until stage T + 1 with remaining capacity $\bar{c}$ as $EV_{t+1}^{\bar{c}}$ in the DSKP formulation, which corresponds to $V_{t+1}(\bar{c})$ in the SSPSP formulation. The optimal policy determines a threshold value for each stage and for each remaining capacity,

$$R_t^{\bar{c}} = EV_{t+1}^{\bar{c}} - EV_{t+1}^{\bar{c}-1},$$

with $R_t^{\bar{c}}$ denoting the *critical reward*. The optimal policy for the particular case of DSKP $\pi^*(t, \bar{c}, r)$ is ACCEPT (REJECT) if $r \geq (<) R_t^{\bar{c}}$.

Proposition 5.1 (McLay et al. 2009): *The optimal policy for SSPSP defines a critical assessed threat value $H_t(\bar{c})$ at each stage t and in each state $s(t) = \bar{c}$, with a passenger whose realized assessed threat value is greater than or equal to this threshold classified as a selectee.*

Theorem 5.2 summarizes several results from Papastavrou et al. (1996) applied to SSPSP. Proposition 5.2 provides two results for SSPSP that can also be applied to the particular case of DSKP with equal weights.

Theorem 5.2 (McLay et al. 2009): *The following results are valid for SSPSP:*
1. $V_t(\bar{c})$ is a concave nondecreasing function of $\bar{c}$, $t = 1,2,\ldots,T$.
2. $V_t(\bar{c})$ is a concave nonincreasing function of t, $\bar{c} = 0,1,\ldots,c$.
3. $H_t(\bar{c})$ is nonincreasing with $\bar{c}$, $t = 1,2,\ldots,T$.
4. $H_t(\bar{c})$ is nonincreasing with t, $\bar{c} = 0,1,\ldots,c$.

Proposition 5.2 (McLay et al. 2009): *The following results are valid for SSPSP:*
1. $V_t(\bar{c})$ is a concave nondecreasing function of p, $\bar{c} = 0,1,\ldots,c$, $t = 1,2,\ldots,T$.
2. $H_t(\bar{c})$ is nondecreasing with p, $\bar{c} = 0,1,\ldots,c$, $t = 1,2,\ldots,T$.

Theorem 5.2 and Proposition 5.2 have several implications for optimal passenger screening. First, the policy becomes more stringent at the end of the time interval, as shown by the critical assessed threat value that is nonincreasing with time, allowing passengers with lower assessed threat values to be classified as selectees. Passengers with lower assessed threat values are more likely to be classified as selectees at the end of the time interval than at the beginning, given that the remaining capacity is the same. Second, the concavity of the expected security with respect to capacity implies that having extra capacity in the selectee class is more beneficial when the remaining capacity is lower. In addition, the expected security increases when more passengers are expected to arrive, which is intuitive since more threats are likely to be detected when more passengers are screened. The concavity of the expected security with respect to p implies that increasing p is more beneficial when p is small. The monotonicity of the critical assessed threat value with respect to p implies that any given passenger is less likely to be classified as a selectee when more passengers are expected to arrive during future stages. The concavity of the critical assessed threat value implies that increasing p when p is large has a more conservative effect on the policy than when p is small. Note that whether a high-risk passenger is classified as a selectee depends on the time that the passenger checks-in and how many passengers have already been identified. In practice, the capacity can be exceeded in order to classify a small number of extra passengers as selectees; the hybrid model presented in Section 5 addresses this point.

McLay et al. (2009) show that SSPSP can be formulated as a particular case of the Generalized SSA (Derman et al. 1972). In SSA, there are n available people for n jobs, with the values of the people given by $v_1, v_2, \ldots, v_n$, which are treated as known constants. The n jobs arrive sequentially, with the values of the jobs being independently and identically distributed random variables $X_1, X_2, \ldots, X_n$ with cumulative distribution function $F(X)$. Therefore, there are n sequential stages in SSA, where in stage t, a job arrives and its value x_t becomes known, $t = 1,2,\ldots,n$. The job is matched to a person who has not yet been assigned to an arriving job. Therefore, job t is assigned to person i_t, $t = 1,2,\ldots,n$. The objective is to maximize the expected total reward, given by the expectation of sum of the products of the values of the job and the person to whom it is matched, $E[\Sigma_{t=1,2,\ldots,n} X_t v_{i_t}]$. Derman et al. (1972) provide the optimal policy for SSA.

GSSA generalizes SSA to consider the number of jobs being different than the number of people. If the number of jobs m is less than the number of people n, then the m people with the highest values are assigned to a job. If $m \geq n$, then m- n pseudo-people are created with value zero. If the probability that a job arrives at each stage is less than one, then the optimal policy can be trivially modified by modeling the event that a job does not arrive as a job arriving with value zero, which can be obtained by modifying $F(X)$. This generalization of SSA to consider a random number of jobs arriving is referred to as the Generalized SSA (GSSA).

McLay et al. (2009) formulates SSPSP as a particular case of GSSA by first creating a set of T people, with c people having value L_S - L_{NS} (corresponding to the spaces available in the selectee class) and T - c people having value zero (corresponding to the spaces available in the nonselectee class). The job arriving at each stage has value zero with probability 1-p, corresponding to no one arriving for check-in during stage t.

With probability p, a job arrives, whose value follows the cumulative density function $F(X)$. The objective function is to maximize the expected additional security from classifying passengers as selectees.

Derman et al. (1972) provide an optimal policy for SSA using dynamic programming. In their approach, in stage t, $T - t + 2$ breakpoints are created using the probability density function of the jobs,

$$-\infty = a_{0,t} \leq a_{1,t} \leq \ldots \leq a_{T-t+1,t} = +\infty$$

where

$$a_{t',t} = \int_{a_{t'-1,t+1}}^{a_{t',t+1}} y\, dF(y) + a_{t'-1,t+1}F(a_{t'-1,t+1}) + a_{t',t+1}(1 - F(a_{t',t+1})).$$

$t = 1,2,\ldots,T-1$, $t' = 1,2,\ldots,T - t + 1$, are computed in advance for all T stages, requiring $O(T^2I)$ time and $O(T^2)$ space (recall that $O(I)$ is the time complexity required to perform integration). When a job arrives during stage t, its realized value x_t falls into one of $T - t + 1$ intervals, with each interval corresponding to one of the remaining available people. Job t is assigned to the person with the i^{th} smallest value who has not been assigned a job such that $a_{i,t} < x_t \leq a_{i+1,t}$. The policy executes in $O(T \log T)$ time, after the breakpoints, $a_{t',t}$, are computed. Note that the optimal policy depends on the distribution of the job values, not the values of the available people.

McLay et al. (2009) show that the optimal policy can be generalized for GSSA by redefining the cumulative density function to take into account the probability of a job not arriving,

$$F_{GSSA}(X) = (1 - p) + pF(X).$$

The optimal policy for GSSA is identical to that of SSA, with $F_{GSSA}(X)$ replacing $F(X)$ in the expression for $a_{t',t}$. Note that the DSKP and GSSA policies are identical, with

$$a_{T-t-\overline{c},t-1}(L_S - L_{NS}) = EV_t^{\overline{c}} - EV_t^{\overline{c}-1} = R_{t-1}^{\overline{c}},$$

$t - 1 = 1,2,\ldots,T$, $\overline{c} = 1,2,\ldots,c$. McLay et al. (2009) also provide the optimal policy for a more general case of GSSA, where the random variables $X_1, X_2, \ldots, X_n$ are not required to be identically distributed.

The breakpoints for GSSA defined by $a_{t',t}$ provide the critical assessed threat values for SSPSP, with

$$H_t(\overline{c}) = a_{T-t-\overline{c}+1,t}.$$

Since the assessed threat values are between zero and one, then $a_{0,t} = 0$ and $a_{T-t+1,t} = 1$, $t = 1,2,\ldots,T$. Proposition 5.3 shows that the optimal way to assign passengers is to classify the highest-risk passengers as selectees. Proposition 5.4 indicates that the optimal policy is insensitive to the security levels.

Proposition 5.3 (McLay et al. 2009): *If the passengers and their assessed threat values are known a priori, then the optimal policy is to classify the c passengers with the highest realized assessed threat values as selectees and the remaining passengers as nonselectees.*

Proposition 5.4 (McLay et al. 2009): *The optimal policy for SSPSP does not depend on the security levels, L_S and L_{NS}.*

McLay et al. (2009) provide an illustrative example for SSPSP. In this example, a total of $T = 2000$ stages are considered, with the probability that a passenger checks-in during each stage given by $p = 0.5$, resulting in 1000 passengers expected to check-in during the time interval. The assessed threat values are assumed to follow a truncated exponential distribution with parameter 1/16. The security level of the selectee and nonselectee classes are $L_S = 0.9$ and $L_{NS} = 0.7$. However, Proposition 5.4 indicates that the optimal policy does not depend on the security levels, and hence, the insights obtained from the analysis are applicable regardless of the real-world security levels. Resolving alarms by secondary screening is not explicitly considered; rather, it is assumed that the capacity of the selectee class is determined by factoring in the expected number of resources needed to resolve false alarms.

Figures 1-3 in McLay et al. (2009) illustrate the value function and critical assessed threat value thresholds that reflect the monotonicity and concavity results in Theorem 5.2 and Proposition 5.2, using a remaining capacity of 10 for illustration purposes. For simplicity, they reflect the expected additional security using the knapsack formulation notation ($EV_t^{\overline{c}}$). Note that these figures do not illustrate how the optimal policy operates in practice (i.e., how the critical assessed threat value actually changes as passengers check in) since Figure 1 illustrates the policy when the remaining capacity and p are fixed, Figure 2 illustrates the optimal policy when the stage and p are fixed, and Figure 3 illustrates the optimal policy when the remaining capacity and the stage are fixed.

The optimal policy was computed using Matlab on a Pentium D 3.2 GHz processor with 3.5 GB of RAM. The capacity of the selectee class considered was $c = 50, 100, 200$ passengers, which is consistent with the observation that most passengers are classified as nonselectees (Barnett 2001). A total of 100 replications were executed for each capacity level. The optimal value functions were computed in advance and were used by all 100 replications. In each replication, a set of passenger arrivals was simulated, and the optimal policy was applied to determine how to screen the passengers as they checked-in. The average objective function values were 46.5, 47.7, and 50.1, for $c = 50, 100, 200$, respectively. These values are within a factor of 0.001 of the optimal screening strategy when all passengers are assumed to be known a priori (see Proposition 5.3), which indicates that the optimal policy is highly effective at classifying high-risk passengers as selectees. These values are absolute expected total security values, and they can be compared to the expected additional security values in Figure 2 by adding a constant factor of $pTL_{NS}E(A(t)) = 43.75$ to the values in Figure 2. Note that the critical assessed threat values were relatively constant until the end of the time interval. Figure 4 in McLay et al. (2009) illustrates the average critical assessed threat values as a function of passenger arrivals, for $c = 50, 100, 200$. When there was no remaining capacity in the selectee class, the critical assessed threat value increased to one until stage T, which increases the average critical assessed threat value and forced any additional passengers to be classified as nonselectees. Figure 4 also illustrates the conditional distributions of the assessed threat values of the passengers who are classified as selectees and nonselectees as the optimal SSPSP policy executed for a single replication. Note that the capacity c decreased and the stage t increased during each replication, whereas c remained constant in Figure 1. It shows that there is very little overlap of the assessed threat values of the passengers classified as selectees and the passengers classified as nonselectees, and hence, the optimal policy was able to consistently identify high-risk passengers.

Once there is no remaining capacity in the selectee class, any additional passengers that check in are classified as nonselectees. This provides a way for passengers to time their arrival at the airport in order to receive less security scrutiny. This issue is compounded by the fact that the optimal SSPSP policy tends to utilize nearly all of the capacity in the selectee class. The entire capacity of the selectee class was used in all but 3, 6, and 11 of the 100 replications, for $c = 50, 100, 200$, respectively. There was only excess capacity when no passengers arrived during the last several stages, and this excess remaining capacity never exceeded two passengers.

To analyze the possibility of passengers arriving at the end of a time interval to avoid being classified as selectees, suppose that there exists a stage t (with $t < T$) at which a passenger was classified as a selectee and effectively used the last selectee class spot. By design, all passengers checking in during stages $t + 1$, $t + 2$, ..., T must be classified as nonselectees. For $c = 50$, 18.1 passengers on average arrive after there is no remaining selectee class capacity, while for $c = 200$, this average drops to 3.4 passengers. Although several passengers arriving at the end of the time interval could avoid being classified as selectees under the optimal SSPSP policy, this affects only a small fraction of the passengers. Moreover, the time intervals can begin and end at arbitrary times, so there is a level of unpredictability in knowing when a passenger can arrive to maximize the likelihood of being classified as a nonselectee, which reduces the ability of terrorists to game the system.

The effectiveness of the optimal SSPSP policy depends on accurately identifying and predicting the assessed threat value distribution and the probability of a passenger arriving during each stage. To study the sensitivity of the optimal SSPSP policy with respect to this distribution, a different assessed threat value distribution of the passengers arriving to check-in was considered and compared to the assessed threat value distribution used to compute the optimal policy. A truncated exponential distribution with parameter 1/16 was used to generate the optimal policy (i.e., the value functions and critical assessed threat values). The passengers arriving for check-in followed truncated exponential distributions with parameters $\lambda = 0.005, 0.01, \ldots, 0.1$. For each such value of λ, 100 replications were executed. Figure 6 in McLay et al. (2009) shows the average total security as a function of λ scaled by the sum of the realized assessed threat values. The scaling was performed since the expected assessed threat value increases with λ, and hence, the total number of threats in the system increased with λ. The optimal objective function value is maximized when $\lambda = 1/16$.

To summarize, SSPSP provides a real-time tool that could be used by the TSA to screen passengers, or for any system where threat object must be detected to protect a secure area. The SSPSP policy has several operational implications if implemented. First, note that the time interval considered by SSPSP can be defined arbitrarily (e.g., one hour), with each stage corresponding to a small time period (e.g., 10 seconds). Therefore, several time intervals would be needed to cover a day, and capacity changes between the time intervals would correspond to changes in security screening personnel staffing and device availability. Passenger arrival rates are variable over the day, but are considered to be constant within each time interval. This is reasonable since the time intervals could be defined to capture shorter periods of time, with different arrival rates in different time intervals. SSPSP treats each time interval independently, but they are not entirely independent in an airport. For example, passengers already waiting in security lines may have been classified as selectees or nonselectees during an earlier time interval than passengers currently arriving at the end of security lines. Independence is less of an issue if the capacity of the selectee class in one time interval depends on the length of the security lines when the time interval begins (e.g., capacity of the selectee class in the next time interval could be reduced if the security lines are long).

One implication of SSPSP is that the optimal policy does not depend on the security levels, L_S and L_{NS}. That is, passengers would be classified the same way if the security level of the selectee class is either nearly the same or much higher than the security level of the nonselectee class. SSPSP nearly always classifies threat passengers that a prescreening system like CAPPS identifies as high-risk as selectees, but CAPPS incorrectly labels some threat passengers as low-risk. It has been observed that even if a prescreening system such as CAPPS is effective at identifying threat passengers, a large proportion of threat passengers may still be classified as nonselectees, because the proportion of passengers classified as selectees is small (Barnett 2004, Martonosi and Barnett 2006). Therefore, it may be of benefit to improve the quality of the screening of nonselectees rather than that of the selectees (i.e., reducing L_S -L_{NS} rather than increasing L_S - L_{NS}).

The optimal SSPSP policy can also be modified to allow for all extremely high-risk passengers as selectees, even when there is no remaining capacity in the selectee lass. In such a hybrid model, let all passengers with an assessed threat value larger than a prespecified threshold θ be classified as selectees, which occurs with probability q = $\int_{\theta}^{1} A(\alpha)d\alpha$, given that a passenger arrives. The objective of this hybrid model is to maximize the expected total security of the passengers who are not automatically classified as selectees (i.e., all passengers whose assessed threat values are less than θ). Then, the original optimality equations and boundary conditions then become a function of θ (see McLay et al. 2009 for details).

There are many criticisms of binary passenger screening systems, as opposed to screening all passengers the same way. In particular, it has been noted that terrorists could probe the system by sending a large number of terrorists through the system prior to carrying out an attack to determine the conditions under which the terrorists would be classified as nonselectees, maximizing the probability of a successful attack (Chakrabarti and Strauss 2002). It is conceivable that in the future, the TSA could abandon binary screening systems to pursue a more uniform screening strategy. However, the challenges and costs associated with deploying next-generation passenger screening technologies at our nation's airports indicate that there are long periods of time when there is not sufficient capacity to screen all aviation passengers with new technologies. For example, the 100% baggage screening mandate requires all checked baggage to be screened by EDSs and explosive trace detection systems (ETDs). In the five years after this measure was enacted (November 19, 2001), over 1400 EDSs and 6600 ETDs were deployed at the nation's commercial airports in order to meet this objective (US GAO 2006). Alternative screening methods such as hand searches and positive passenger baggage matching were used to screen checked baggage for explosives until EDSs and ETDs became available. Deploying new security screening technologies (temporarily) creates a binary passenger screening system, even when the intention is to screen all passengers with the new technology. Therefore, SSPSP remains highly relevant even if uniform screening methods are pursued, and its analysis can be used to address questions of how to optimally use limited resources in next-generation passenger screening systems.

In conclusion, McLay et al. (2009) introduces SSPSP, which models stochastic passenger and baggage screening systems. SSPSP, which considers passengers arriving over time, is formulated as a Markov Decision Process, where the optimal policy can be computed by dynamic programming. The solution to SSPSP provides a real-time passenger screening methodology, which suggests that SSA could be part of a tool used by the TSA for screening passengers in real-time. In addition, the relationship between SSPSP, DSKP, and SSA is made explicit, illustrating several theoretical properties of the optimal policy.

An illustrative example was analyzed to study the optimal SSPSP policy. Analysis of the solutions suggests that extremely high-risk passengers are almost certainly classified as selectees, regardless of when these passengers check in. Critical assessed threat values were relatively constant over the time interval. The optimal policy was shown to be more sensitive to the accuracy of the assessed threat value distribution and less sensitive to the probability of passengers checking-in during each stage. Several implications of implementing such a policy are discussed. One implication is that the optimal policy does not depend on the security levels, which implies that it may be of benefit to improve the quality of the screening of nonselectees rather than just selectees, which currently receive the most security attention and consideration.

The optimal policy for SSPSP is highly effective in classifying extremely high-risk passengers as selectees. Exceptions occur at the very end of the time intervals and when the assessed threat value distribution is inaccurate. The fact that a small number of high-risk passengers could be classified as nonselectees could be a potential weakness of SSPSP. However, in practice, extremely high-risk passengers would not be classified as nonselectees, even if there is no remaining capacity in the selectee class. For security purposes, it is reasonable to classify one extra passenger as a selectee and have the security lines move slightly slower, and the hybrid model as discussed in McLay et al. (2009) can be used to mitigate some of this risk. Therefore, the possibility of terrorists gaming the system by timing their check-ins can be circumvented.

SSPSP addresses the direct effects of passenger screening, such as detecting threat items, not the indirect effects of passenger screening, such as deterrence. Although the implementation of SSPSP does not depend on the security levels, the amount of deterrence is likely to depend on the security levels, and in particular, L_S

-L_{NS} (Martonosi and Barnett 2006). One extension to this research is the development of models that capture the indirect effects of the security levels on system security.

There are several other possible directions to extend the research results presented. First, considering screening costs can give insight into the design of cost-effective screening strategies. Second, understanding how the optimal policy can be manipulated by terrorists wishing to be screened by less secure classes would be of interest to the aviation security policy community. In particular, modeling how terrorists could take advantage of how optimal policies operate in real-time as well as manipulating the passenger pool (e.g., flooding the airport with decoys) may provide the impetus for designing robust heuristics and algorithms. Work is in progress to address all these extensions.

6. Statistical Tests to Empirically Compare Tabu Search Parameters

The application of stochastic heuristics, like tabu search or simulated annealing, to address hard discrete optimization problems has been an important advance for efficiently obtaining good solutions in a reasonable amount of computing time. A challenge when applying such heuristics is assessing when a particular set of parameter values yields better performance compared to other such sets of parameter values. For example, it can be difficult to determine the optimal mix of memory types to incorporate into tabu search. This in turn prompts users to undertake a trial and error phase to determine the best combination of parameter settings for the problem under study. Moreover, for a given problem instance, one set of heuristic parameters settings may yield a better solution than another set of parameters, for a given initial solution. However, the performance of this heuristic on this instance for a single heuristic execution is not sufficient to assert that the first set of parameters settings will always produce superior results than the second set of parameters, for all initial solutions.

Jacobson and McLay (2009) look at three known statistical procedures (one of which is a basic statistical test procedure and two of which were developed for discrete event simulation (discrete) optimization) to assess and compare the computational performance of tabu search for MAX 3-SATISFIABILITY. The statistical procedures designed for application within the domain of discrete event simulation output analysis (a paired difference t-test and two multiple comparison procedures developed and studied by B.L. Nelson and others) are adapted for this new purpose. An empirical case study is reported by computationally studying MAX 3-SATISFIABILITY instances across thirty-two variations of tabu search.

Discrete optimization problems are defined by a finite set of solutions together with an objective function value assigned to each solution (Garey and Johnson 1979). The goal when addressing such problems is to find solutions that globally optimize the objective function. Unless otherwise noted, assume that discrete optimization problems are minimization problems.

Computational complexity theory provides a well-defined framework to assess the difficulty of these problems. Garey and Johnson (1979) formally defines the classes *P* (Polynomial), *NP* (Non-deterministic Polynomial) and *NP-complete*. The class *P* contains decision problems that can be solved in polynomial time in the size of the problem. The class *NP* contains decision problems for which it can be checked in polynomial time (in the size of the problem) whether a given potential solution solves the problem. The hardest problems in the class *NP* are categorized as *NP-complete*. The seminal paper by Cook (1971) proves the first decision problem, SATISFIABILITY, to be *NP-complete*. Over the past 30+ years, a significant amount of research has been done in this area, with numerous problems proven to be *NP-complete*. See Garey and Johnson (1979) for an in-depth discussion on the theory of *NP-completeness*, including an extensive list of *NP-complete* problems.

Numerous heuristics have been developed to find near-optimal solutions to *NP-complete* problems using a reasonable amount of computing time. General search strategies such as *simulated annealing* (Henderson et al. 2003), *genetic algorithms* (Reeves 2003), *tabu search* (Glover and Laguna 1997), *threshold accepting* (Dueck and Scheuer 1990) and the *noising method* (Charon and Hudry 2001), each use a distinct search rule to intelligently exploit the solution space with the hope of finding optimal/near-optimal solutions. One difficulty when applying such heuristics is being able to assess when a set of parameter values lead to good performance. For example, it can be difficult to determine the optimal mix of memory types to incorporate into tabu search. This in turn prompts users of such heuristics to undertake an initial trial and error phase to determine the best combination of parameter settings for the problem under study. However, given the stochastic nature of such heuristics, it is not clear how the performance of a heuristic using different parameter settings can be compared in a systematic way. For example, for a given problem instance, one set of heuristic parameters settings may yield a better solution than another set of parameters, for a given initial solution. However, the performance of this heuristic on this instance for a single heuristic execution is not sufficient to assert that the first set of parameters settings will always produce superior results than the second set of parameters, for all initial solutions. Adenso-Diaz and Laguna (2006) address this issue by creating a support tool to determine the optimal set of search parameters values for heuristics using Taguchi's fractional factorial experimental designs and local search procedures.

This paper is a case study on three known statistical procedures that can be used to assess the computational effectiveness of stochastic heuristics for hard discrete optimization problems. The statistical procedures includes paired difference t-tests, single-stage and two-stage multiple comparisons procedures. These procedures are computationally illustrated on different variations of tabu search applied to a set of MAX 3-SATISFIABILITY instances. Although the results are illustrated on one class of *NP-complete* problems (namely, MAX 3-SATISFIABILITY) using one heuristic (namely, tabu search), the statistical procedures in this paper can be applied to other hard discrete optimization problems and heuristics to assess the performance of sets of value parameters. To describe SATISFIABILITY, several definitions are needed. A *clause* is a combination of r *literals* where a literal is a Boolean variable (x_i) or its negation ($\bar{x}_i$). A solution, ω, is a Boolean vector of size n, where each Boolean variable is assigned a value of one or zero. Given a solution, a clause is satisfied if at least one literal in this clause takes on the value one. Using these definitions, SATISFIABILITY can now be described: Given a collection of m clauses, involving n Boolean variables, is there a solution (i.e., values for the n Boolean variables) that satisfies all m clauses?

Several variations of SATISFIABILITY have been studied, based on the number of literals assigned to each clause. For example, if the number of literals is r for all clauses, then SATISFIABILITY is referred to as r-SATISFIABILITY (e.g., 2-SATISFIABILITY, 3-SATISFIABILITY).

By definition, SATISFIABILITY is a decision problem, since the solution yields either a "yes" or "no" response (i.e., all m clauses can or cannot be satisfied). On the other hand, any SATISFIABILITY instance can be formulated as an optimization problem (termed MAX SATISFIABILITY) by introducing the objective function

$$\text{Max } g(\omega) = \Sigma_{j=1,2,\dots,m} C_j(\omega) \qquad \text{or, equivalently} \qquad \text{Min } f(\omega) = \Sigma_{j=1,2,\dots,m} (1-C_j(\omega)),$$

where $C_j(\omega) = 1$ (0) if clause j is (not) satisfied for solution ω. SATISFIABILITY was the first problem proven to be *NP-Complete* (Cook 1971). SATISFIABILITY is also fundamental for solving several real-world problems, ranging from computer chip design to robotics (e.g., see Gu et al. 1996 for an extensive list of application areas). Such applications have been the driving force behind much of the research done to address the problem.

To describe the general framework for tabu search for discrete optimization problems, define Ω to be a finite set of feasible solutions (solution space). Define an objective function f: $\Omega \to Z^n$ that maps elements of the solution space into an n-tuple of integer values. Note that the range of this function is often just Z, the set of integer scalars. Using these definitions, a discrete optimization problem can be represented as: $\text{Min}_{\omega\in\Omega} f(\omega)$.

Tabu search (Glover and Laguna 1997) is a heuristic for addressing a wide variety of intractable discrete optimization problems, ranging from minimizing makespan in a flow shop problem, to planning and re-planning in a project schedule problem (Garbowski and Pempera 2007, Calhoun et al. 2002). Hybrid heuristics that use tabu lists have also been developed for machine scheduling problems and hub location problems (Chen and Wu 2006, Chen 2007).

To describe tabu search, several definitions are needed. Define a *neighborhood function*, η:$\Omega \rightarrow 2^{\Omega}$, where $\eta(\omega) \subseteq \Omega$ for all $\omega \in \Omega$. Define the *tabu list* $TL \subset \Omega$ to be a set of elements identified as forbidden (tabu) solutions. TL is initially empty, and subsequently updated by prespecified rules (tabu conditions) that use historical information from the search. Define *BEST* to be an evaluator function that finds the best solution among a set of neighboring solutions $\eta(\omega)\backslash TL$. Let ω^* denote the best solution found to date and define a stopping condition which terminates the algorithm (e.g., a desired number of iterations has elapsed or all neighboring solutions are tabu). Using these definitions and notations, pseudo-code for a generic tabu search algorithm is presented (Glover and Laguna 1997).

```
1. Select an initial solution $\omega \in \Omega$
   $\omega^* \leftarrow \omega$ and set the iteration counter $k = 0$ and $TL = \emptyset$
2. $k \leftarrow k+1$
   Select $\omega_k \in \eta(\omega)\backslash TL$ such that $\omega_k = BEST(\eta(\omega)\backslash TL)$
   $\omega \leftarrow \omega_k$
   If $f(\omega) < f(\omega^*)$, then $\omega^* \leftarrow \omega$
3. If the stopping condition is met, then STOP and report $\omega^*$.
   Otherwise, update $TL$ and return to Step 2.
```

The tabu list is a memory-based structure that guides the search away from undesirable solutions (e.g., local optima or previously visited solutions) by defining a set of tabu solutions. Tabu search traverses the solution space by selecting the best possible solution at each step, determined by the function *BEST*. Note that the best possible solution might not always yield an improvement in the objective function value. In such cases, tabu search allows the search to select an inferior solution (with a minimum deterioration in the objective function value). The function *BEST* also gives tabu search an ability to use other methods (e.g., linear programming) to make the search more flexible.

There are a wide range of tabu search strategies that have been developed, including long-term memory, and aspiration criteria (see Glover and Laguna 1997 for a list of other strategies). In the tabu search framework, *TL* is usually used as a short-term memory tool that forbids a solution to be visited over the next L iterations. After L iterations have occurred, the same solution can be repeated since its tabu status has been removed. It is not desirable for an algorithm to revisit the same solution too frequently. To avoid such a situation, it may be necessary to incorporate long-term memory into the algorithm, which uses frequency information of previously visited solutions and guides the search into unexplored regions of the solution space by avoiding solutions that have already been visited a large number of times. In some cases, the simultaneous application of short-term and long-term memories may result in the neighboring solutions becoming too restricted. To circumvent this problem, aspiration criteria may be defined for overriding the tabu status of a solution (e.g., a tabu solution produces a better solution than the best solution found to date, ω^*). The types of memory and the tabu strategies used are all important factors that affect the performance of tabu search. These factors need to be individually designed for each specific problem.

Several definitions are needed to describe tabu search for MAX 3-SATISFIABILITY. Let n denote the number of Boolean variables and m denote the number of clauses for a given instance of MAX 3-SATISFIABILITY. Let Ω be the set of all possible (2^n) solutions (i.e., the solution space), where every solution $\omega \in \Omega$ is represented as a binary vector of size n. Define an objective function, $f\colon \Omega \rightarrow [0,m]$, such that $f(\omega) = \Sigma_{j=1,2,\ldots,m} (1 - C_j(\omega))$ denotes the number of *unsatisfied* clauses associated with solution $\omega \in \Omega$. Note that by minimizing the number of unsatisfied clauses rather than maximizing the number of satisfied clauses, the problem being addressed is MIN 3-UNSATISFIABILITY rather than MAX 3-SATISFIABILITY. However, by design, there is a one-to-one mapping between the solutions of these two problems, and hence, solving one problem equivalently solves the other problem.

Let $\omega_k \in \Omega$ denote the current solution (at iteration k). A neighboring solution, $\omega_k(i) \in \eta(\omega_k)$, is obtained by flipping the value of the Boolean variable $x(i)$ in ω_k (e.g., neighboring solution $\omega_k(50)$ is obtained by changing $x(50)$ from 0(1) to 1(0) in ω_k). This switching procedure (often referred to as 1-flip; see Yagiura and Ibaraki 2001) defines the neighboring function. There are exactly n distinct neighboring solutions $\{\omega_k(1), \omega_k(2), \ldots, \omega_k(\mathrm{n})\}$ associated with the current solution ω_k. For tabu search, define the set $\eta(\omega_k)\backslash T$, to be all candidate neighboring solutions (i.e., the set of all neighboring solutions that are not tabu). Define $\omega_{k,\mathrm{best}} \in \operatorname{argmin}\{f(\omega_k(i)), \omega_k(i) \notin T, i = 1,2,\ldots,n\}$ to be the solution with the smallest (best) objective function value among all candidate neighboring solutions obtained from ω_k.

Yagiura and Ibaraki (2001) present an excellent computational study with tabu search and other heuristics applied to MAX SATISFIABILITY that compares different r-flip neighborhoods, while Smyth et al. (2003) describe an iterated robust tabu search for MAX SATISFIABILITY. Gu (1993) describes general local search algorithms for satisfiability problems. Since the focus of this paper is not to identify optimal tabu search algorithms for MAX SATISFIABILITY, but rather, to describe how statistical procedures can be used to compare the effectiveness of different tabu search variations, commonly used tabu search variations are considered here.

Three basic types of tabu search strategies (short-term tabu list, long-term tabu list, aspiration criteria) are applied to MAX 3-SATISFIABILITY. A neighboring solution $\omega_k(i)$ is defined as tabu if the Boolean variable $x(i)$, which is to be switched in ω_k to obtain $\omega_k(i)$, is either on the short-term tabu list, Υ, or the long-term tabu list, Γ. To avoid excessive memory requirements, both lists are defined as a set of Boolean variables rather than a set of solutions. It should be noted that both the short-term and long-term tabu lists are dynamic and therefore proper updating procedures need to be defined. In particular, at the end of iteration k, Boolean variable $x_{\mathrm{best}}(i)$, which is switched to obtain $\omega_{k,\mathrm{best}}$, immediately enters the short-term tabu list and stays tabu for the next L iterations (i.e., $x_{\mathrm{best}}(i)$ leaves the list at the end of iteration $k + L$). Note that any Boolean variable $x(i)$ may re-enter the short-term tabu list later in the algorithm's execution. If a Boolean variable $x(i)$ enters the short-term tabu list more than F times, it is then included on the long-term tabu list. Once a Boolean variable is included on the long-term tabu list, it stays tabu for the rest of the algorithm's execution.

Two n-tuple vectors, u and v, store all the information required to verify the tabu status of the Boolean variables. When a Boolean variable $x(i)$ enters the short-term tabu list, $u(i)$ records the current iteration counter (i.e., $u(i) = k$) and $v(i)$ records the number of times that the Boolean variable $x(i)$ has become tabu (i.e., $v(i) = v(i) + 1$). The same updating process is repeated at all iterations $k = 1,2,\ldots,K$. These two vectors makes it possible to check the tabu status of any Boolean variable x(i) with just two comparisons (i.e., $x(i) \in \Upsilon$ if $k \leq u(i) + L$ and $x(i) \in \Gamma$ if $v(i) \geq F$). As noted earlier, a neighboring solution $\omega_k(i)$ is said to be tabu if the Boolean variable $x(i)$ is tabu (i.e., $x(i) \in \Upsilon \cup \Gamma$). However if a tabu neighboring solution $\omega_k(i)$ produces a better solution than the best solution found to date (i.e., $f(\omega_k(i)) < f(\omega^*)$) then its tabu status is overridden: this defines a simple aspiration criterion for the algorithm.

Tabu search is initialized with a randomly generated initial solution, ω_l (its source of randomness), and empty tabu lists, Υ and Γ. A set of n neighboring solutions $\{\omega_l(1), \omega_l(2), \ldots, \omega_l(n)\}$ are obtained, where for each such solution $\omega_l(i)$, $f(\omega_l(i))$ is evaluated, its short-term and long-term tabu status (i.e., $x(i) \in \Upsilon \cup \Gamma$) is checked, and the solutions ω^* and $\omega_{l,\text{best}}$ are updated. Note that during the first iteration, no neighboring solutions are tabu, since both tabu lists are initialized to be empty. Once the objective function value and tabu status of all n neighboring solutions are obtained, the solution $\omega_{l,\text{best}}$ becomes the current solution for the next iteration, $k = 2$ (even if it is worse than the current solution ω_l) and the Boolean variable $x_{\text{best}}(i)$ enters the short-term tabu list. This process is repeated until K iterations have been executed, all neighboring solutions have become tabu, or a solution satisfying all m clauses is found. The algorithm is then terminated and the best solution found to date, ω^*, is reported. Tabu search for MAX 3-SATISFIABILITY is outlined in pseudo-code form, where short-term tabu list, long-term tabu list, and a simple aspiration criterion are applied as described in the Section 4.2.

```
Generate a random initial solution $\omega_l$ and evaluate $f(\omega_l)$
Set $\omega^* \leftarrow \omega_l, f(\omega^*) \leftarrow f(\omega_l)$
Initialize the tabu lists, $\Upsilon = \emptyset$, $\Gamma = \emptyset$, and set the iteration counter $k = 1$
Repeat
    Set the Move Indicator MOVE = NO and set the Boolean variable index $i = 1$
    Repeat
        Obtain a neighboring solution $\omega_k(i) \in \eta(\omega_k)$ and evaluate $f(\omega_k(i))$
        If $f(\omega_k(i)) = 0$, then STOP and report $\omega_k(i)$ (i.e., all m clauses are satisfied)
        If $f(\omega_k(i)) < f(\omega^*)$, then set MOVE = YES, $\omega^* \leftarrow \omega_k(i)$, $f(\omega^*) \leftarrow f(\omega_k(i))$,
                $\omega_{k,\text{best}} \leftarrow \omega_k(i), f(\omega_{k,\text{best}}) \leftarrow f(\omega_k(i))$
        If $x(i) \notin (\Upsilon \cup \Gamma)$
            If MOVE = NO, then set MOVE = YES, $\omega_{k,\text{best}} \leftarrow \omega_k(i), f(\omega_{k,\text{best}}) \leftarrow f(\omega_k(i))$
            If $f(\omega_k(i)) < f(\omega_{k,\text{best}})$, then $\omega_{k,\text{best}} \leftarrow \omega_k(i), f(\omega_{k,\text{best}}) \leftarrow f(\omega_k(i))$
        $i \leftarrow i+1$
    Until $i = n$
    If MOVE = NO, then STOP and report $\omega^*, f(\omega^*)$ (i.e., all neighboring solutions are tabu)
    Set $\omega_{k+1} \leftarrow \omega_{k,\text{best}}, f(\omega_{k+1}) \leftarrow f(\omega_{k,\text{best}})$
    Update the tabu lists $\Upsilon, \Gamma$ and set $k \leftarrow k+1$
Until $k = K$
Report $\omega^*, f(\omega^*)$
```

Three statistical procedures are applied to compare and assess the performance of heuristics for hard discrete optimization problems. The first procedure is a paired difference t-test. The second and third procedures are single-stage and two-stage multiple comparisons procedures, respectively, adapted from discrete event simulation optimization output analysis (Banks et al. 2001). There are several common links between discrete optimization problem heuristics and discrete event simulation output analysis procedure. Jacobson and Yucesan (1999) establish that the difficulty of several decision problems associated with discrete event simulation modeling and analysis can be studied using computational complexity. They show that accessibility of states, ordering of events, noninterchangeability of model implementations, and execution stalling for discrete event simulation can be formulated as NP-hard search problems. Jacobson and Yucesan (2002) also identify common issues between the two areas. They define NEIGHBORHOOD, which seeks a sequence of events such that two distinct states can be accessed, one state after executing all but the last event and another state after executing all the events, and INITIALIZE, which seeks a sequence of events such that executing the sequence with one particular initial event results in a particular state being reached, while for a second initial event, that particular state cannot be reached, and proves them to be NP-hard.

The three procedures are illustrated on thirty-two tabu search variations applied to four instances of MAX 3-SATISFIABILITY taken from SATLIB (Hoos and Stützle 2000). Two of these instances (UF200-0100, with 200 variables and 860 clauses, and UF225-0100, with 225 variables and 960 clauses) are satisfiable, while two of the instances (UUF125-0100, with 125 variables and 538 clauses, and UUF175-0100, with 175 variables and 753 clauses) are unsatisfiable (with the optimal solutions each having one unsatisfied clause).

The tabu length, L, denotes the size of the short-term tabu list (i.e., $L = |\Upsilon|$). The tabu frequency, F, denotes the maximum number of times a Boolean variable can be inserted on the short-term tabu list. If a Boolean variable attempts to enter the short-term tabu list more than F times, then this Boolean variable is included on the long-term tabu list, Γ, and stays tabu for the rest of the algorithm's execution. As L is increased or F is decreased, candidate neighboring solutions (i.e., $\eta(\omega) \setminus (\Upsilon \cup \Gamma)$) become more restricted, since the tabu lists will contain more Boolean variables. If the aspiration criterion, A, is used, then the restriction on candidate neighboring solutions is relaxed, since the aspiration criterion overrides the tabu status of a Boolean variable.

Thirty-two tabu search variations were considered for each of the four MAX 3-SATISFIABILITY instances, with each tabu search variation experiment executed for thirty replications of 1000 iterations each (where each of these replications were defined by a set of independent and identically distributed initial solutions). These thirty-two variations are obtained by considering all combinations of four short-term tabu lengths (L = 5, 10, 25, 50), four long-term tabu frequencies (F = 5, 10, 25, 50), and the use of aspiration criteria (A = Yes, No). Instances UF200-0100 and UF225-0100 took 12 CPU minutes and 15 CPU minutes while instances UUF125-0100 and UUF175-0100 took 5 CPU minutes and 9 CPU minutes for all thirty replications (independent of the tabu search parameter values) on a DELL OPTIPLEX GX 270 with Intel Pentium IV Processor, 2.66 GHz clock speed with 1.0 GB of RAM implemented in Microsoft Visual C++ 6.0.

To statistically compare the effectiveness of the thirty-two tabu search variations, three different statistical procedures were implemented. Table 1 in Jacobson and McLay (2009) reports the best and worst objective function values for each of the four instances, over the thirty-two tabu search parameter setting (L, F, and A) combinations that resulted in these values, obtained across all thirty replications. Note that when more than one set of parameters are reported, this indicates that all these tabu search parameters resulted in the best or worst solution reported. Also, for UF225-0100, none of the tabu search variations was able to find an optimal solution (several parameter setting resulted in solutions with one unsatisfied clause). Table 1, column (i) represents the tabu search parameters associated with the smallest of the best objective function values across the thirty-two tabu search variations and thirty replications, for each of the four problem instances, while column (ii) represents the tabu search parameters associated with the largest of the best objective function (i.e., the worst of the best) values across the thirty-two tabu search variations and thirty replications, for each of the four problem instances.

To identify which tabu search variation was most effective for the four problem instances, paired difference t-tests were conducted (Walpole et. al. 2002), where the results from the mean best (smallest) objective value found across thirty replications for each tabu search algorithm variation were paired to assess statistical differences between the four short-term tabu lengths (L = 5, 10, 25, 50), the four long-term tabu frequencies (F = 5, 10, 25, 50), and the use of aspiration criteria (A = Yes, No). The pairing was done such that all the tabu search parameter values matched except for the parameter being tested. All the null hypotheses posed were that the two variations under study yielded the same results, versus the appropriate one-sided alternative hypothesis. For example, from Table 2 in Jacobson and McLay (2009), for problem UF200-0100, H_0: $\mu_{A=\text{YES}} = \mu_{A=\text{NO}}$ with H_A: $\mu_{A=\text{YES}} < \mu_{A=\text{NO}}$ resulted in the test statistic value T = 2.235 distributed Student-t with 15 degrees of freedom (sixteen paired average data values, obtained from the mean best objective function value across the thirty replications for thirty-two tabu search algorithm variations), with a corresponding p-value of 0.021. Therefore, the null hypothesis is rejected in favor of the alternative hypothesis for any size of the test less than this p-value. For the short-term memory parameter F, six paired difference t-tests were obtained, each involving eight paired data values (eight paired average data values, obtained from the mean best objective function value across the thirty replications for sixteen tabu search algorithm variations), hence the resulting test statistics had 7 degrees of freedom. For long-term memory, six paired difference t-tests were obtained, each involving eight paired data values (eight paired average data values, obtained from the mean best objective function value across the thirty replications for sixteen tabu search algorithm variations), hence the resulting test statistics had 7 degrees of freedom. The full set of results with paired difference t-tests for instances UF200-0100 and UUF175-0100 are reported in Tables 2 (Jacobson and McLay report a similar table for instances UF225-0100 and UUF125-0100). Jacobson and McLay (2007) also report the mean ($\bar{f}$) and the standard deviation (s) (computed across the thirty replications) of the best (smallest) objective function values found during each of the algorithm executions, for each tabu search variation.

The paired difference results suggest that using an aspiration criterion results in statistically significant superior results compared to not using an aspiration criterion for all four tabu problem instances. The average best objective function values for experiments conducted with an aspiration criterion are smaller than for those without an aspiration criterion, which is consistent with the results obtained with the paired difference t-tests. For short-term memory, the parameter value L = 25 resulted in statistically significant superior results compared with L = 5, 10 and 50. For long-term memory, the parameter values F = 10 and F = 25 result in statistically significant superior results compared with F = 5 and 50. Note that since all the testing is done pairwise, conclusions drawn from this analysis that have implications across all the tabu search algorithm variations are affected by the Bonferroni inequality (Hochberg and Tamhane 1987). However, since the total number of hypotheses tested is small, and many of the p-values are less than .001, the resulting effect on the conclusions drawn from the analysis will be limited.

Using the same data sets to analyze the different tabu search variations results in positive dependency across these variations. Unfortunately, it would be difficult to assess or measure this dependency explicitly. However, to account for the effect of this dependency across the tabu search algorithm variations, Nelson (1992) describes a single-stage multiple comparisons procedure for discrete event simulation optimization that can be adapted to determine the most effective tabu search variations from amongst the thirty-two tabu search variations implemented. In particular, for any two tabu search variations, labeled i and j, based on n (= 30) replications, the standard deviation can be estimated by

$$\hat{\sigma}_{ij}(n) = (S_i^2(n) + S_j^2(n) - 2S_{ij}(n))^{1/2}$$

where $S_i^2(n)$ is the sample variance for tabu search variation i obtained over the n replications and $S_{ij}(n)$ is the sample covariance between tabu search variations i and j. This expression takes into account the data dependency across the tabu search variations, as noted above. From Jacobson and McLay (2007), the tabu search parameters L = 25, F = 25, A = YES yielded the overall best tabu search performance for UF200-0100 and UF225-0100, and the tabu search parameters L = 25, F = 50, A = YES yielded the best tabu search performance among variations using the aspiration criterion for UUF125-0100 and UUF175-0100. For the corresponding instances, label these tabu search algorithm variations TS1. Therefore, the estimator $\hat{\sigma}_{ij}(n)$ was only computed for the fifteen pairs of tabu search variations with an aspiration criterion (since tabu search with an aspiration criterion yielded better results than without an aspiration criterion).

For each of the fifteen tabu search variation with the aspiration criterion (denoted by the parameters $L,F,$YES($=A$), the null hypotheses H$_0$: $\mu_{\text{TS1}} = \mu_{L,F,\text{YES}}$ is rejected in favor of the one-sided alternative hypothesis H$_A$: $\mu_{\text{TS1}} < \mu_{L,F,\text{YES}}$ at the α level of the test if

$$\bar{f}_{\text{TS1}} + t_{(1-\alpha),\,n-1}\,(\hat{\sigma}_{TS1j}(n)/\sqrt{n}) < \bar{f}_{L,F,YES}. \qquad (6.1)$$

Tables 3 in Jacobson and McLay (2009) reports the level of the test at which the null hypotheses H$_0$: $\mu_{\text{TS1}} = \mu_{L,F,A}$ would be rejected by the alternative hypothesis H$_A$: $\mu_{\text{TS1}} < \mu_{L,F,A}$ for the fifteen tabu search variations with the aspiration criterion, for instances UF200-0100 and UUF175-0100 (Jacobson and McLay 2007 report similar results for instances UF225-0100 and UUF125-0100). For all four problem instances, for ten or more of the fifteen tabu search variations, the null hypotheses were rejected in favor of the alternative hypotheses at a level of test less than .001. Once again, the Bonferroni inequality and the simultaneous testing of fifteen tabu search variations mean that the effective level for these tests is lower than what is reported. However, given the level at which each of these tabu search variations are rejected, the effective level of the tests for the fifteen rejected tabu search variations all remain close to zero.

To further analyze and compare the tabu search variations that were not rejected for each problem instance in addition to the best tabu search variation TS1 (between three and six total tabu search variations), paired difference t-tests were used to determine if there was any statistically significant difference between them. Thirty additional replications of each tabu search variation were executed using the same initial solutions for each problem instance. Therefore, for each problem instance, replication i = 1,2,...,30 had the same initial solution for each tabu search variation. The t-tests compared the difference between the best objective function value in each of the thirty replications between two tabu search variations. Therefore, for each paired difference t-test, there were thirty paired data values between two tabu search variations, resulting in 29 degrees of freedom. These results are reported in Table 4 in Jacobson and McLay (2009) for instances UF200-0100 and UUF175-0100 (Jacobson and McLay 2007 report similar results for instances UF225-0100 and UUF125-0100).

Nelson and Matejcik (1995) describe a simultaneous ranking, selection, and multiple comparisons with the best two-stage procedure to determine the best design across a discrete set of designs in steady state discrete event simulation. Similar to the Nelson single-stage procedure, the structure of this procedure also makes it adaptable to assess the effectiveness of different variations of a heuristic like tabu search for discrete optimization problems. Jacobson and McLay (2009) provide a comprehensive description of the Nelson and Matejcik two-stage procedure within the context of assessing the performance of tabu search variations.

In the first stage of the Nelson and Matejcik two-stage procedure, a set of independent tabu search variation replications are executed. The best solutions obtained across these variations are reported and used to compute the sample variance of the difference, denoted by S^2. The number additional replications for each tabu search variation required to obtain the desired coverage level are determined. These additional replications are executed and the resulting sample means (over all replications) for each tabu search variation are reported and used to obtain multiple comparison with the best confidence intervals, which are used to assess and identify those tabu search variations that yield the best results.

For most problems, it would seem that this normality condition would be difficult, if not impossible, to satisfy or to verify. However, for some discrete optimization problems, the objective functions are computed as a sum or average, hence by the central limit theorem, the normality assumption may indeed hold. For example, the MAX 3-SATISFIABILITY objective function is the sum of m clauses, while the objective function for the traveling salesman problem is the sum of distances between cities. Therefore, although this assumption may appear untenable to satisfy or to verify, there are problems where it may be reasonable.

The Nelson and Matejcik two-stage procedure assumes that the unknown variance-covariance matrix exhibits a structure known as *sphericity* (Nelson and Matejcik 1995). For the application here, this would mean the variances of all pairwise best objective function value differences across tabu search variations are equal, even though the marginal variances and covariances may be unequal. This suggests a weak type of uniform covariance structure across tabu search variations. Note that it would be impossible to verify such a result in practice. However, Nelson and Matejcik (1995) show that their procedure is extremely robust to departures from sphericity, provided the covariances are positive, which is likely to be the case here since tabu search variations all seek to optimize the objective function. They further note that the assumption of sphericity may not be exactly or even approximately satisfied in many situations, and hence, suggest that it may be prudent to slightly inflate the nominal coverage probability to ensure adequate coverage. Given these observations, the choice of $\alpha = .05$ used below may result in a slight lower coverage probability. Goldsman and Nelson (1998) provide a complete description of the Nelson and Matejcik two-stage procedure. Note that the value of $T^{(1-\alpha)}_{k-1,(k-1)(n_0-1),0.50}$ in Step 1 (see Goldsman and Nelson 1998) can be obtained from Table 4 in Hochberg and Tamhane (1987) or Table B.3 in Bechhofer et al. (1995), while values that fall outside of the tables can be computed using a FORTRAN computer program (see Dunnett 1989).

Table 5 in Jacobson and McLay (2009) reports results with the Nelson and Matejcik two-stage procedure applied to instances UF200-0100 and UUF175-0100 (Jacobson and McLay 2007 reports similar results for instances UF225-0100 and UUF125-0100). For UF200-0100, N was computed to be 79, while for UUF175-0100, N was computed to be 65 (both for $\delta = 0.75$, $\alpha = 0.05$, $n_0 = 30$, $g = 2.57$). For the same parameter values, with $\delta = 0.75$, N was computed to be 45 and 31, respectively. By design, N-30 additional replications were executed for each of the instances. The results reported in Table 5 suggest that an appropriate choice of parameters can help to identify those tabu search variations that are statistically indistinguishable. The results obtained are all consistent with the conclusions drawn from the Nelson single-stage procedure.

It would be interesting to further study other ways to use statistical analysis tools within this domain, hence provide a well-defined framework for comparing and evaluating the performance of heuristics. Similar statistical tests can be used to assess determine the optimal settings for heuristics for MAX 3-SATISFIABILITY as well as for other hard discrete optimization problems. Moreover, using methods such as common random numbers and antithetic random numbers may provide more insights into the effectiveness of such heuristics.

7. Other Research Results

In addition to the results reported above,, several other results were obtained during the period of the grant. These results are briefly discussed here.

In the area of aviation security system design and analysis, McLay et al. (2008) examines selective checked baggage screening systems that use a prescreening system and two types of baggage screening devices, one to screen checked baggage of passengers perceived as lower-risk and the other to screen checked baggage of passengers perceived as higher-risk. A cost-benefit analysis of such selective checked baggage screening systems is reported. The analysis is performed for several scenarios that consider various levels of accuracy of prescreening systems in assessing passenger risk. The results indicate that the accuracy of the prescreening system in assessing passenger risk is more important for reducing the number of successful attacks than the effectiveness of the checked baggage screening devices at detecting threats when few passengers are classified as higher-risk. Moreover, several selective screening scenarios are identified that may be preferable to current checked baggage screening strategies. Lee et al. (2008) surveys operations research activities directed toward analyzing and enhancing the aviation security system, with the intent to be informative and motivate the continual improvement of our nation's transportation security.

In the area of scheduling problems and algorithm analysis, Kao et al. (2009) present a modified Branch and Bound (B&B) algorithm called, the Branch, Bound and Remember (BB&R) algorithm, which uses the Distributed Best First Search (DBFS) exploration strategy for solving the $1| r_i | \Sigma t_i$ scheduling problem, a single machine scheduling problem where the objective is to find a schedule with the minimum total tardiness. Memory-based dominance strategies are incorporated into the BB&R algorithm. In addition, a modified memory-based dynamic programming algorithm is also introduced to efficiently compute lower bounds for the $1| r_i | \Sigma t_i$. Computational results are reported, which shows that the BB&R algorithm with the DBFS exploration strategy outperforms the best known algorithms reported in the literature.

In the area of asset allocation problems, the primary application of such research has been to vaccine formulary design optimization. Hall et al. (2008a) formulate an integer programming model that solves for the maximum number of vaccines that can be administered without any extraimmunization for pediatric immunization. An exact dynamic programming algorithm and a randomized heuristic for the integer programming model is also formulated and the heuristic is shown to be a harmonic approximation algorithm. Computational results are reported on three sets of test problems, based on existing and future childhood immunization schedules, to demonstrate their computational effectiveness and limitations. Given that future childhood immunization schedules may need to be solved for each child, on a case-by-case basis, the results reported here may provide a practical and valuable tool for the public health community. Hall et al. (2008b) introduce the General Vaccine Formulary Selection Problem (GVFSP) to model general childhood immunization schedules that may be used to illuminate these alternatives and choices by selecting a vaccine formulary that minimizes the cost of fully immunizing a child and the amount of extraimmunization. Both exact algorithms and heuristics for GVFSP are presented. A computational comparison of these algorithms and heuristics is presented for the Recommended Childhood Immunization Schedule, as well as several randomly generated childhood immunization schedules that are likely to be representative of future childhood immunization schedules. The results reported here provide both fundamental insights into the structure of the GVFSP models and algorithms, as well as practical value for the public health community.

References

E. Aarts, L.K. Lenstra, (1997) *Local Search in Combinatorial Optimization*. Wiley and Sons, New York.

B. Adenso-Diaz, M. Laguna, (2006) Fine-Tuning of Algorithms using Fractional Experimental Designs and Local Search. *Operations Research*, 54(1), 99-114.

D.E. Armstrong, S.H. Jacobson (2003), Studying the Complexity of Global Verification for NP-hard Discrete Optimization Problems, *Journal of Global Optimization,* 27(1), 83-96.

D.E. Armstrong, S.H. Jacobson (2005), "Data Independent Neighborhood Functions and Strict Local Optima," *Discrete Applied Mathematics*, 146(3), 233-243.

D.E. Armstrong, S.H. Jacobson (2006a), "Order Preserving Reductions and Polynomial Improving Paths," *Operations Research Letters*, 34(1), 9-16.

D.E. Armstrong, S.H. Jacobson (2006b), "Analyzing the Complexity of Finding Good Neighborhood Functions for Local Search Algorithms," *Journal of Global Optimization*, 36(2), 219-236.

D.E. Armstrong, S.H. Jacobson (2008), "An Analysis of Neighborhood Functions on Generic Solution Spaces," *European Journal of Operational Research,* 186(2), 529-541.

R. K. Ahuja, O. Ergum, J. B. Orlin, A.P. Punnen (2002), "A Survey of Very Large-Scale Neighborhood Search Techniques," *Discrete Applied Mathematics*, 123(1-3), 75-102.

E. Angel, V. Zissimopoulos (2000), "On the Classification of NP-complete Problems in Terms of Their Correlation Coefficient," *Discrete Applied Mathematics*, 99, 261-277.

V. L. Babu, R. Batta, L. Lin (2006), "Passenger Grouping Under Constant Threat Probability in an Airport Security System," *European Journal of Operational Research,* 168, 633-644.

J. Banks, J.S. Carson, B.L. Nelson, D.M. Nicol, (2001) *Discrete Event System Simulation*, Third Edition, Prentice Hall Inc., Upper Saddle River, New Jersey.

R.E. Bechhofer, T.J. Santner, D.M. Goldsman, (1995) Design and Analysis of Experiments for Statistical Selection, Screening, and Multiple Comparisons. John Wiley and Sons, New York.

A. Barnett (2001), "The Worst Day Ever," *OR/MS Today*, 28(6), 28-31.

A. Barnett (2004), "CAPPS II: The Foundation of Aviation Security? *Risk Analysis*, 24(4), 909-916.

A. Barnett, M. Abraham, V. Schimmel (1979), "Airline Safety: Some Empirical Findings," *Management Science*, 25, 1045-1056.

A. Barnett, M. K. Higgins (1989), "Airline Safety: The Last Decade," *Management Science*, 35, 1-21.

A. Barnett, R.W. Shumsky, M. Hansen, A. Odoni, G. Gosling (2001), "Safe at Home? An Experiment in Domestic Airline Security," *Operations Research*, 49, 181-195.

P. Billingsley (1979), *Probability and Measure*, John Wiley and Sons, New York.

E.K. Burke, B.L. MacCarthy, S. Petrovic, R. Qu (2003), "Knowledge Discovery in a Hyper-heuristic for Course Timetabling Using Case-based Reasoning," *Lecture Notes in Computer Science*, 2740, 276-287.

V. Butler, R.W. Poole Jr. (2002), "Rethinking Checked-Baggage Screening," Reason Public Policy Institute, Policy Study No. 297. Los Angeles, CA.

K.M. Calhoun, R.F. Deckro, J.T. Moore, J.W. Chrissis, J.C. Van Hove (2002) "Planning and Re-planning in Project and Production Scheduling," *Omega*, 30(3), 155-170.

S. Chakrabarti, A. Strauss (2002), "Carnival Booth: An Algorithm for Defeating the Computer-Aided Passenger Screening System," First Monday, 7 (available at www.¯rstmonday.org).

I. Charon, O.. Hudry (2001), "The Noising Methods: A Generalization of Some Metaheuristics," *European Journal of Operational Research* 135(1), 86-101.

G. Chartrand, O.R. Oellermann (1993), *Applied and Algorithmic Graph Theory*, McGraw-Hill, Inc., New York.

J.-F. Chen (2007), "A Hybrid Heuristic for the Uncapacitated Single Allocation Hub Location Problem," *Omega*, 35(2), 211-220.

J.-F. Chen, T.-H. Wu, (2006), "Total Tardiness Minimization on Unrelated Parallel Machine Scheduling with Auxiliary Equipment Constraints," *Omega*, 34(1), 81-89.

C.A. Coello, D.A. Van Veldhuizen, G.B. Lamont (2002), *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, New York.

S.A. Cook (1971) The Complexity of Theorem-Proving Procedures, in *Proceedings of the Third Annual ACM Symposium on Theory of Computing, Association for Computing Machinery*, New York, 151-158.

I. Das (1999), "A Preference Ordering Among Various Pareto Optimal Alternatives," *Structural Optimization*, 18(1), 30-35.

C. Derman, G.J. Lieberman, S.M. Ross (1972), "A Sequential Stochastic Assignment Problem," *Management Science*, 18(7), 349-355.

G. Dueck, T. Scheuer (1990) "Threshold Accepting: A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing," *Journal of Computational Physics* 90, 161-175.

C.W. Dunnett (1989) "Multivariate Normal Probability Integrals with Product Correlation Structure," *Applied Statistics* 38, 564-579 (Correction: 42, 709).

M. Ehrgott, X. Gandibleux (eds.) (2002), *Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys*, Kluwer Academic Publishers, Boston.

FAA, Federal Aviation Administration (2006), FAA Aerospace Forecasts: Fiscal Years 2006-2017. Office of Policy and Plans, Federal Aviation Administration, United States Department of Transportation, Washington, DC.

M.A. Fleischer, S.H. Jacobson (1999), "Information Theory and the Finite-Time Behavior of the Simulated Annealing Algorithm: Experimental Results," *INFORMS Journal on Computing*, 11(1), 35-43.

J. Garbowski, J. Pempera (2007), "The Permutation Flow Shop Problem with Blocking: A Tabu Search Approach," *Omega*, 35(3), 302-311.

M. R. Garey, D.S. Johnson (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York.

F. Glover, M. Laguna (1997), *Tabu Search*, Kluwer Academic Publishing, Norwell, MA.

D. Goldsman, B.L. Nelson, (1998), "Statistical Screening, Selection, and Multiple Comparison Procedures in Computer Simulation," in *Proceedings of the 1998 Winter Simulation Conference* (D.J. Medeiros, E.F. Watson, J.S. Carson, and M.S. Manivannan, Editors). IEEE. Piscataway, New Jersey, pp 159-166.

J. Gu (1993), "Local Search for Satisfiability (SAT) Problem," *IEEE Transactions on Systems, Man, and Cybernetics*, 23(4), 1108-1129.

J. Gu, P.W. Purdom, J. Franco, B.W. Wah (1996), "Algorithms for the Satisfiability Problem: A Survey," in DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 35: Satisfiability Problem: Theory and Applications: Proceedings of a DIMACS Workshop, March 11-13, 1996, (D. Du, J. Gu and P.M. Pardalos, Editors), 19-130.

M. Hall, D. DeLollis, July 14, 2004, "Plan to Collect Flier Data Canceled," *USA Today* 1.

S.N. Hall, E.C. Sewell, S.H. Jacobson (2008a), "Maximizing the Effectiveness of a Pediatric Vaccine Formulary While Prohibiting Extraimmunization, *Health Care Management Science*, 11(4), 339-352.

S.N. Hall, S.H. Jacobson, E.C. Sewell (2008b), "An Analysis of Pediatric Vaccine Formulary Selection Problems," *Operations Research*, 56(6), 1348-1365.

F. Harary, E.M. Palmer (1973), *Graphical Enumeration*, Academic Press, Inc., New York.

G.H. Hardy, J.E. Littlewood, G. Polya (1952), *Inequalities* (2nd ed.), Cambridge University Press, Cambridge, England.

K. Helsgaun (2000), "An Effective Implementation of the Lin-Kernighan Traveling Salesman Problem Heuristic," *European Journal of Operational Research*, 126, 106-130.

D. Henderson, S.H. Jacobson, A.W. Johnson (2003), "The Theory and Practice of Simulated Annealing," 287-319, (Chapter 10 in *Handbook on Metaheuristics*, F. Glover and G. Kochenberger, Editors), Kluwer Academic Publishers, Norwell, Massachusetts.

Y. Hochberg, A.C. Tamhane (1987), *Multiple Comparison Procedures*, John Wiley and Sons, New York.

H.H. Hoos, T. Stützle, (2000) SATLIB: An Online Resource for Research on SAT, in SAT2000, I.P. Gent, Maaren, H.V., T. Walsh, Editors, IOS Press, pp 283-292.

S.H. Jacobson, E. Yucesan (1999), "On the Complexity of Verifying Structural Properties of Discrete Event Simulation Models," *Operations Research*, 47(3), 476-481.

S.H. Jacobson, E. Yucesan (2002),"Common Issues in Discrete-Event Simulation and Discrete Optimization," *IEEE Transactions on Automatic Control*, 47(2), 341-345.

S.H. Jacobson, L.A. McLay (2007), "An Empirical Study of Tabu Search Applied to MAX 3-SATISFIABILITY," Technical Report, Department of Computer Science, University of Illinois, Urbana, Illinois.

S.H. Jacobson, J.M. Bowman, J.E. Kobza (2001), "Modeling and Analyzing the Performance of Aviation

Security Systems using Baggage Value Performance Measures," *IMA Journal of Management Mathematics*, 12, 3-22.

S.H. Jacobson, S.N. Hall, L.A. McLay, J.E. Orosz (2005), "Performance Analysis of Cyclic Simulated Annealing Algorithms," *Methodology and Computing in Applied Probability*, 7(2), 183-201.

S.H. Jacobson, L.A. McLay (2009), "Applying Statistical Tests to Empirically Compare Tabu Search Parameters for MAX 3-SATISFIABILITY: A Case Study," *OMEGA*, 37(3), 522-534.

S.H. Jacobson, L.A. McLay, J.E. Kobza, J.M. Bowman (2005a), "Modeling and Analyzing Multiple Station Baggage Screening Security System Performance," *Naval Research Logistics*, 52(1), 30-45.

S.H. Jacobson, L.A. McLay, J.E. Virta, J.E. Kobza (2005b), "Integer Program Models for the Deployment of Airport Baggage Screening Security Devices," *Optimization and Engineering*, 6(3), 339-359.

Nikolaev, A.G., Jacobson, S.H., Hall, S.N., Henderson, D., 2010, "A Framework for Analyzing Sub-Optimal Performance of Local Search Algorithm", *Computational Optimization and Applications* (to appear).

S.H. Jacobson, D. Solow (1993), "The Effectiveness of Finite Improvement Algorithms for Finding Global Optima," *Methods and Models of Operations Research*, 37, 257-272.

S.H. Jacobson, K.A. Sullivan, A.W. Johnson (1998), "Discrete Manufacturing Process Design Optimization Using Computer Simulation and Generalized Hill Climbing Algorithms," *Engineering Optimization*, 31, 247-260.

S.H. Jacobson, J.E. Virta, J.M. Bowman, J. E. Kobza, J.J., Nestor (2003), "Modeling Aviation Baggage Screening Security Systems: A Case Study," *IIE Transactions*, 35, 259-269.

D.S. Johnson, C.H. Papadimitriou, M. Yannakakis (1988), "How Easy is Local Search?" *Journal of Computers and System Science*, 37, 79-100.

E. Kahn (2006), "Dynamic Risk Analysis System for Aviation Security," *Proceedings of the 4th International Aviation Security Technology Symposium*, Washington, D.C.

E. Kahn, R. Robinson (2006), "Risk Management Analysis for US Commercial Aviation Security," *Proceedings of the 4th International Aviation Security Technology Symposium*, Washington, D.C.

Kao, G.K., Sewell, E.C., Jacobson, S.H., 2009, "A Branch, Bound, and Remember Algorithm for the $1|r_i|\Sigma t_i$ Scheduling Problem," *Journal of Scheduling*, 12(2), 163-175.

G. Kao, S.H. Jacobson, 2008, "Finding Preferred Subset of Pareto Optimal Solutions," *Computational Optimization and Applications*, 40(1), 73-95.

E.M. Kasprzak, K.E. Lewis (2001), "Pareto Analysis in Multiobjective Optimization Using the Collinearity Theorem and Scaling Method," *Structural and Multidisciplinary Optimization*, 22(3), 208-218.

S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi (1983), "Optimization by Simulated Annealing," *Science*, 220(4598), 671-680.

J.E. Kobza, S.H. Jacobson (1996), "Addressing the Dependency Problem in Access Security System Architecture Design," *Risk Analysis*, 16(6), 801-812.

J.E. Kobza, S.H. Jacobson (1997), "Probability Models for Access Security System Architectures," *Journal of the Operational Research Society*, 48(3), 255-263.

P. Korhonen, M. Halme (1990), "Supporting the Decision Maker to Find the Most Preferred Solutions for a MOLP-Problem, in *Proc. of the 9th Int. Conf. on Multiple Criteria Decision Making*, Fairfax, Virginia, 173-183.

A.J. Lee, L.A McLay, S.H. Jacobson, S.H. (2009), "Designing Aviation Security Passenger Screening Systems using Nonlinear Control," *SIAM Journal on Control and Optimization*, 48(4), 2085-2105.

A.J. Lee, A.G. Nikolaev, S.H. Jacobson (2008), "Protecting Air Transportation: A Survey of Operations Research Applications to Aviation Security," *Journal of Transportation Security*, 1(3), 160-184.

S. Lin, B.W. Kernighan (1973), "An Effective Heuristic for the Traveling Salesman Problem," *Operations Research*, 21, 498-516.

LKH (2005), LKH Version 1.3 (July 2002). Retrieved Nov 7, 2005 from http://www.akira.ruc.dk /~keld/research/LKH.

J.E. Marsden (1987), *Basic Complex Analysis*, W.H. Freeman and Company, New York.

S.E. Martonosi, A.I. Barnett (2006), "How Effective is Security Screening of Airline Passengers?" *Interfaces*, 36(6), 545-552.

S.E. Martonosi (2005), "An Operations Research Approach to Aviation Security," Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA.

C.A. Mattson, A.A. Mulur, A. Messac (2004), "Smart Pareto Filter: Obtaining a Minimal Representation of Multiobjective Design Space," *Engineering Optimization*, 36(6), 721-740.

L. A. McLay, S.H. Jacobson (2007a), "Integer Knapsack Problems with Set-up Weights," *Computational Optimization and Applications*, 37(1), 35-47.

L.A. McLay, S.H. Jacobson (2007b), "Algorithms for the Bounded Set-Up Knapsack Problem," *Discrete Optimization*, 4(2), 206-212.

L.A. McLay, S.H. Jacobson, J.E. Kobza (2005), "Making Skies Safer: Applying Operations Research to Aviation Passenger Prescreening Systems," *OR/MS Today*, 32(5), 24-31.

L.A. McLay, S.H. Jacobson, J.E. Kobza (2007), "Integer Programming models and Analysis for a Multilevel Passenger Screening Problem," *IIE Transactions*, 39(1), 73-81.

L.A. McLay, S.H. Jacobson, J.E. Kobza (2006), "A Multilevel Passenger Prescreening Problem for Aviation Security," *Naval Research Logistics*, 53(3), 183-197.

L.A. McLay (2006), Designing Aviation Security Systems: Theory and Practice, Ph.D. Thesis, University of Illinois at Urbana-Champaign, Urbana, IL.

L.A. McLay, S.H. Jacobson, A.G. Nikolaev, A.G. (2009), "A Sequential Stochastic Passenger Screening Problem for Aviation Security," *IIE Transactions*, 41(6), 575-591

L.A. McLay, S.H. Jacobson, J.E. Kobza (2008), "The Tradeoff between Technology and Prescreening Intelligence in Checked Baggage Screening for Aviation Security," *Journal of Transportation Security*, 1(2), 107-126.

K.M. Mead (2002), "Key Challenges Facing the Transportation Security Administration," Office of the Inspector General, Report Number CC-2002-180, Department of Transportation, Washington, DC.

K.M. Mead (2003), "Aviation Security Costs," Transportation Security Administration. Office of the Inspector General, Report Number CC-2003-066, Department of Transportation, Washington, DC.

A. Messac, C.A. Mattson (2004), "Normal Constraint Method with Guarantee of Even Representation of Complete Pareto Frontier," *AIAA Journal*, 42(10), 2101-2111.

A. Messac, A. Ismail-Yahaya, C.A. Mattson (2003), "The Normalized Normal Constraint Method for Generating the Pareto Frontier," *Structural and Multidisciplinary Optimization*, 25(2), 86-98.

K. M. Miettinen (1999), *Nonlinear Multiobjective Optimization*, Kluwer Academic Publishers, Boston.

K.M. Miettinen, M.M. Makela (2000), "Interactive Multiobjective Opimization System WWW-NIMBUS on the Internet," *Computers and Operations Research*, 27, 709-723.

B.L. Nelson (1992), "Designing Efficient Simulation Experiments," in *Proceedings of the 1992 Winter Simulation Conference*, Arlington, Virginia, 126-132.

B.L. Nelson, F.J. Matejcik (1995), "Using Common Random Numbers for Indifference-Zone Selection and Multiple Comparisons in Simulation," *Management Science*, 41(12), 1935-1945.

X. Nie, R. Batta, C.G. Drury, L. Lin (2006), "Passenger Grouping with Risk Levels in an Airport Security System," Technical Report, University of Buffalo, Buffalo, NY.

A.G. Nikolaev, S.H. Jacobson, L.A. McLay (2007), "A Sequential Stochastic Security System Design Problem for Aviation Security," *Transportation Science*, 41(2), 182-194.

A.G. Nikolaev, S.H. Jacobson (2009), "Using Markov Chains to Analyze the Effectiveness of Local Search Algorithms," Technical Report, Department of Computer Science, University of Illinois, Urbana, Illinois.

J.E. Orosz, S.H. Jacobson (2002), "Finite-time Performance Analysis of Static Simulated Annealing Algorithms," *Computational Optimization and Applications*, 21(1), 21-53.

J.D. Papastavrou, S. Rajagopalan, A.J. Kleywegt (1996), "The Dynamic and Stochastic Knapsack Problem with Deadlines," *Management Science*, 42(12), 1706-1718.

R. Peeters (2003), "The Maximum Edge Biclique Problem is NP-complete," *Discrete Applied Mathematics*, 131, 651-654.

R.W. Poole Jr., G. Passantino (2003), "A Risk-Based Airport Security Policy," Reason Public Policy Institute, Policy Study No. 308, Los Angeles, CA.

M.L. Puterman, 1994, Markov Decision Processes: Discrete Stochastic Dynamic Programming, New York, NY, John Wiley & Sons, Inc.

C. Reeves (2003), "Genetic Algorithms", Chapter 3 in Handbook of Metaheuristics, Kluwer Academic Publishing, Norwell, Massachusetts.

G. Reinelt (1991), "TSPLIB - A Traveling Salesman Problem Library," *ORSA Journal on Computing*, 3(4), 376-385.

V. Rodl, C.A. Tovey (1987), "Multiple Optima in Local Search," *Journal of Algorithms*, 8, 250-259.

S. Savage, P. Weiner, A. Bagchi (1976), "Neighborhood Search Algorithms for Guaranteeing Optimal Traveling Salesman Tours Must be Inefficient," *Journal of Computers and System Science*, 12, 25-35.

R. Singer (2004), "Life After Death for CAPPS II?" *Wired News*. http://www.wired.com.

K. Smyth, H.H. Hoos, T. Stutzle, (2003), "Iterated Robust Tabu Search for MAX_SAT," Advances in Artificial Intelligence, Proceedings, Lecture Notes in Artificial Intelligence, 2671, 129-144.

C.A. Tovey (1985), "Hill Climbing with Multiple Local Optima," *SIAM Journal of Algebraic Discrete Methods*, 6, 384-393.

TSA (2004), U.S. Department of Homeland Security, TSA-Rev. 8-23-2004. http://www.tsa.gov.

TSA (2005), The National Archives and Records Administration File, FR Document 12405. http://www.tsa.gov.

USGAO (2006), United States Government Accountability Office (July 2006), "Transportation Security

Administration: Oversight of Explosive Detection Systems Maintenance Contracts can be Strengthened," Technical Report GAO-06-795, Washington, DC.

USGAO (2005), United States Government Accountability Office (March 2005), "Secure Flight Development and Testing Under Way, but Risks Should be Managed as System is Further Developed," Technical Report GAO-05-356, Washington, DC.

J.E. Virta, S.H. Jacobson, J.E. Kobza (2002), "Outgoing Selectee Rates at Hub Airports," *Reliability Engineering and System Safety*, 76(2), 155-165.

V.G. Vizing (1977), "Complexity of the Traveling Salesman Problem in the Class of Monotonic Improvement Algorithms," *Cybernetics*, 13, 623-626.

R.E. Walpole, R.H. Myers, S.L. Myers, K. Ye (2002), *Probability and Statistics for Engineers and Scientists*, Seventh Edition, Prentice Hall, Upper Saddle River, New Jersey.

M. Yagiura, T. Ibaraki (2001), "Efficient 2 and 3-flip Neighborhood Search Algorithms for the MAX SAT: Experimental Evaluation," *Journal of Heuristics* 7 (5), 423-442.

CONTRIBUTING PERSONNEL

The principal investigator for this project, Sheldon H. Jacobson, Ph.D., has devoted both academic year time and summer time in each of the three years of this project. Derek E. Armstrong, Ph.D., Laura A. McLay, Ph.D., Major Shane N. Hall, USAF, Ph.D., Alex G. Nikolaev, Ph.D., Gio K. Kio, Ph.D., and Adrian J. Lee, Ph.D. were former Ph.D. students of the principal investigator, who worked on aspects of this project. Major Matthew (JD) Robbins, USAF, Douglas M. King, and David Morrison, are current graduate students of the principal investigator who has also contributed to various aspects of this project.

TRANSITIONS

Research on sequential stochastic assignment modeling and analysis has been communicated and shared with personnel in the Department of Homeland Security (Contact: John J. Nestor, Ph.D.).

PUBLICATIONS

The following is a list of published papers in refereed journals or book chapters that are related to the research effort supported in whole or in part by this grant.

Armstrong, D.E., Jacobson, S.H., 2008, "An Analysis of Neighborhood Functions on Generic Solution Spaces," *European Journal of Operational Research*, 186(2), 529-541.

Hall, S.N., Sewell, E.C., Jacobson, S.H., 2008, "Maximizing the Effectiveness of a Pediatric Vaccine Formulary While Prohibiting Extraimmunization, *Health Care Management Science*, 11(4), 339-352.

Hall, S.N., Jacobson, S.H., Sewell, E.C., 2008, "An Analysis of Pediatric Vaccine Formulary Selection Problems," *Operations Research*, 56(6), 1348-1365.

Hall, S.N., Jacobson, S.H., Sewell, E.C., 2008, "Optimizing Pediatric Vaccine Formularies," Chapter 4 in *Optimization in Medicine and Biology* (G. J. Lim, E. K. Lee, Editors) Auerbach Publications (Taylor and Francis Group), 117-145.

Jacobson, S.H., McLay, L.A., 2009, "Applying Statistical Tests to Empirically Compare Tabu Search Parameters for MAX 3-SATISFIABILITY: A Case Study," *OMEGA*, 37(3), 522-534.

Jacobson, S.H., Lee, A.J., Nikolaev, A.G., 2009, "Designing for Flexibility in Aviation Security Systems," *Journal of Transportation Security*, 2(1&2), 1-8.

Kao, G., Jacobson, S.H., 2008, "Finding Preferred Subset of Pareto Optimal Solutions," *Computational Optimization and Applications*, 40(1), 73-95.

Kao, G.K., Sewell, E.C., Jacobson, S.H., 2009, "A Branch, Bound, and Remember Algorithm for the $1|r_i|\Sigma t_i$ Scheduling Problem," *Journal of Scheduling*, 12(2), 163-175.

Lee, A.J., Nikolaev, A.G., Jacobson, S.H., 2008, "Protecting Air Transportation: A Survey of Operations Research Applications to Aviation Security," *Journal of Transportation Security*, 1(3), 160-184.

Lee, A.J., Nikolaev, A.G., Jacobson, S.H., Nestor, J.J., 2008, "Operations Research Applications in Aviation Security Systems," Volume 2, Chapter 8, in *Aviation Security Management* (A. R. Thomas, Editor), 126-145.

Lee, A.J., McLay, L.A., Jacobson, S.H., 2009, "Designing Aviation Security Passenger Screening Systems using Nonlinear Control," *SIAM Journal on Control and Optimization*, 48(4), 2085-2105.

McLay, L.A., Jacobson, S.H., Nikolaev, A.G., 2009, "A Sequential Stochastic Passenger Screening Problem for Aviation Security," *IIE Transactions*, 41(6), 575-591

McLay, L.A., Jacobson, S.H., Kobza, J.E., 2008, "The Tradeoff between Technology and Prescreening Intelligence in Checked Baggage Screening for Aviation Security," *Journal of Transportation Security*,

1(2), 107-126.

Nikolaev, A.G., Jacobson, S.H., McLay, L.A., 2007, "A Sequential Stochastic Security System Design Problem for Aviation Security," *Transportation Science*, 41(2), 182-194.

HONORS/AWARDS

Sheldon H. Jacobson, Ph.D., was named a *Willett Faculty Scholar in the College of Engineering* at the University of Illinois for the 2005-2009 academic years. This designation is given to tenured faculty who are excelling in their contributions to the university. Approximately twenty faculty have been awarded this distinction, based on their record of research accomplishments and achievements.

Sheldon H. Jacobson and Laura A. McLay, Ph.D., were awarded the *2009 Outstanding IIE Publication Award* by the Institute of Industrial Engineers, for research that was supported by the AFOSR. This award is the highest IIE honor for an outstanding original publication that has appeared in any IIE-sponsored or co-sponsored medium. The award is made to the author or authors of that published work, which is judged to be a meritorious contribution to the profession of industrial engineering.

Adrian J Lee, Ph.D., was awarded an Honorable Mention in the *2009 INFORMS Transportation Science & Logistics Dissertation Prize Competition*, for research supported by the AFOSR.

Adrian J. Lee, Ph.D., was recognized nationally for his research supported by the AFOSR when he was awarded a *Graduate Research Award in the Program on Public-Sector Aviation Issues*, sponsored by the Federal Aviation Administration and U.S. Department of Transportation. Dr. Lee was selected as one of ten graduate students nationally to receive their 2008-2009 award. The results of this research will be presented at the Transportation Research Board's Annual Meeting in Washington, D.C. in January 2010.